US005941965A

United States Patent [19]

Moroz et al.

[11] Patent Number: 5,941,965

[45] Date of Patent: Aug. 24, 1999

[54] UNIVERSAL DOCKING STATION

[75] Inventors: **John A. Moroz**, Plymouth; **Gary A. Altenberg**, Buffalo; **James Ternus**, Maple Grove, all of Minn.

[73] Assignee: **Electronics Accessory Specialists International, Inc.**, Scottsdale, Ariz.

[21] Appl. No.: **08/679,131**

[22] Filed: **Jul. 12, 1996**

Related U.S. Application Data

[60] Provisional application No. 60/017,725, May 16, 1996.

[51] **Int. Cl.**[6] **G06F 13/00**

[52] **U.S. Cl.** **710/101**; 361/683

[58] **Field of Search** 395/281, 552, 395/555; 364/708.1; 361/683–686

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,941,845	7/1990	Eppley et al. .	
4,969,830	11/1990	Daly et al. .	
5,187,645	2/1993	Spalding et al. .	
5,373,149	12/1994	Rasmussen .	
5,452,180	9/1995	Register et al. .	
5,477,415	12/1995	Mitcham et al. .	
5,632,020	5/1997	Gephardt et al.	395/283
5,634,080	5/1997	Kikinis et al.	395/893
5,724,529	3/1998	Smith et al.	395/309

OTHER PUBLICATIONS

Don Anderson et al., *CardBus System Architecture*, 150–153, 194–201, 228–231; 322–325 (1995).

Don Anderson, *PCMCIA System Architecture*, 16–Bit PC Cards, 2nd Edition, 146–167, 214–215, 218–225, 296–297 (1995).

Brochure entitled "Card Station Expanding Your Portable World", Axonix Corporation (1994).

Primary Examiner—Glenn A. Auve

Attorney, Agent, or Firm—Merchant, Gould, Smith, Edell, Welter & Schmidt, P.A.

[57] **ABSTRACT**

A universal docking station for coupling a portable computer to a plurality of peripheral devices via a PCMCIA interface is provided, wherein at least one of the peripheral devices is a user input device. The docking station comprises a PCMCIA interface adapted to be coupled to the portable computer, peripheral control units adapted to be coupled to the plurality of peripheral devices, and a docking station control unit coupled to the PCMCIA interface and the peripheral control units, wherein the docking station control unit selectively transfers data between the PCMCIA interface and one of the peripheral control units, and wherein the docking station control unit extends the period of time data written by the computer remains stable on the PCMCIA interface. A computer system including the universal docking station is also provided. A method for interfacing a standard peripheral device to a computer via a PCMCIA bus is also provided.

5 Claims, 6 Drawing Sheets

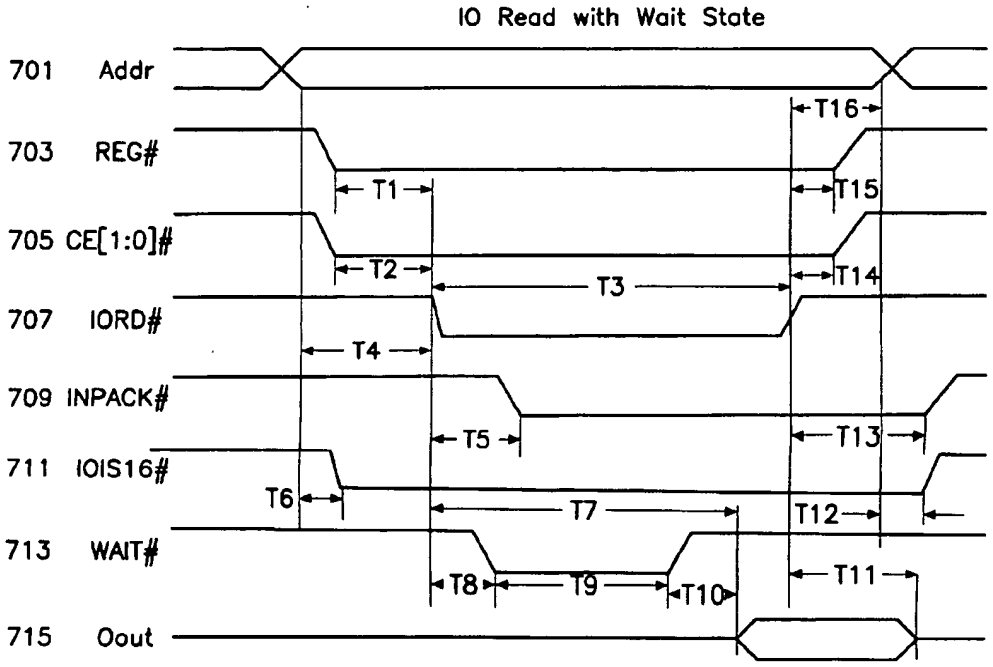
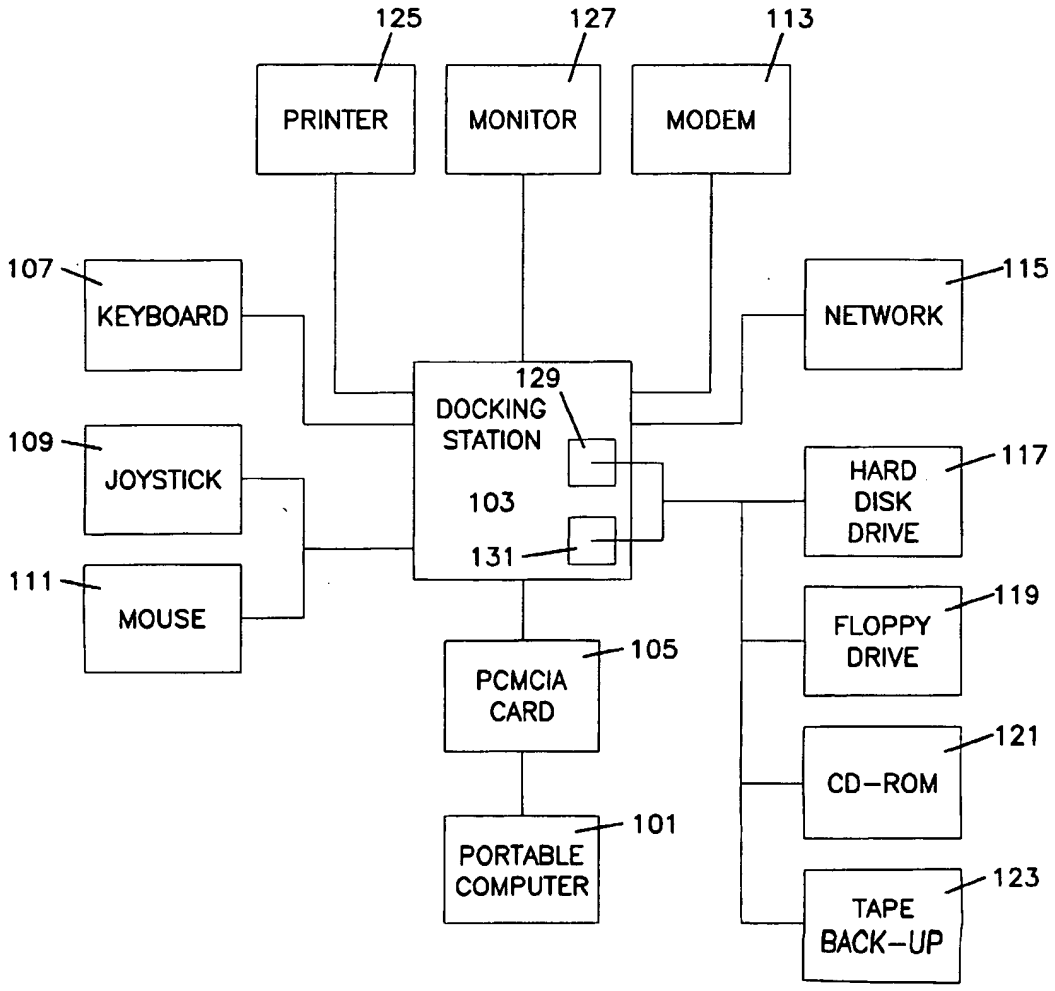


FIG. 1


125
127
113
PRINTER
MONITOR
MODEM
107
KEYBOARD
115
NETWORK
129
DOCKING STATION
103
131
109
JOYSTICK
117
HARD DISK DRIVE
111
MOUSE
119
FLOPPY DRIVE
105
PCMCIA CARD
121
CD-ROM
101
PORTABLE COMPUTER
123
TAPE BACK-UP

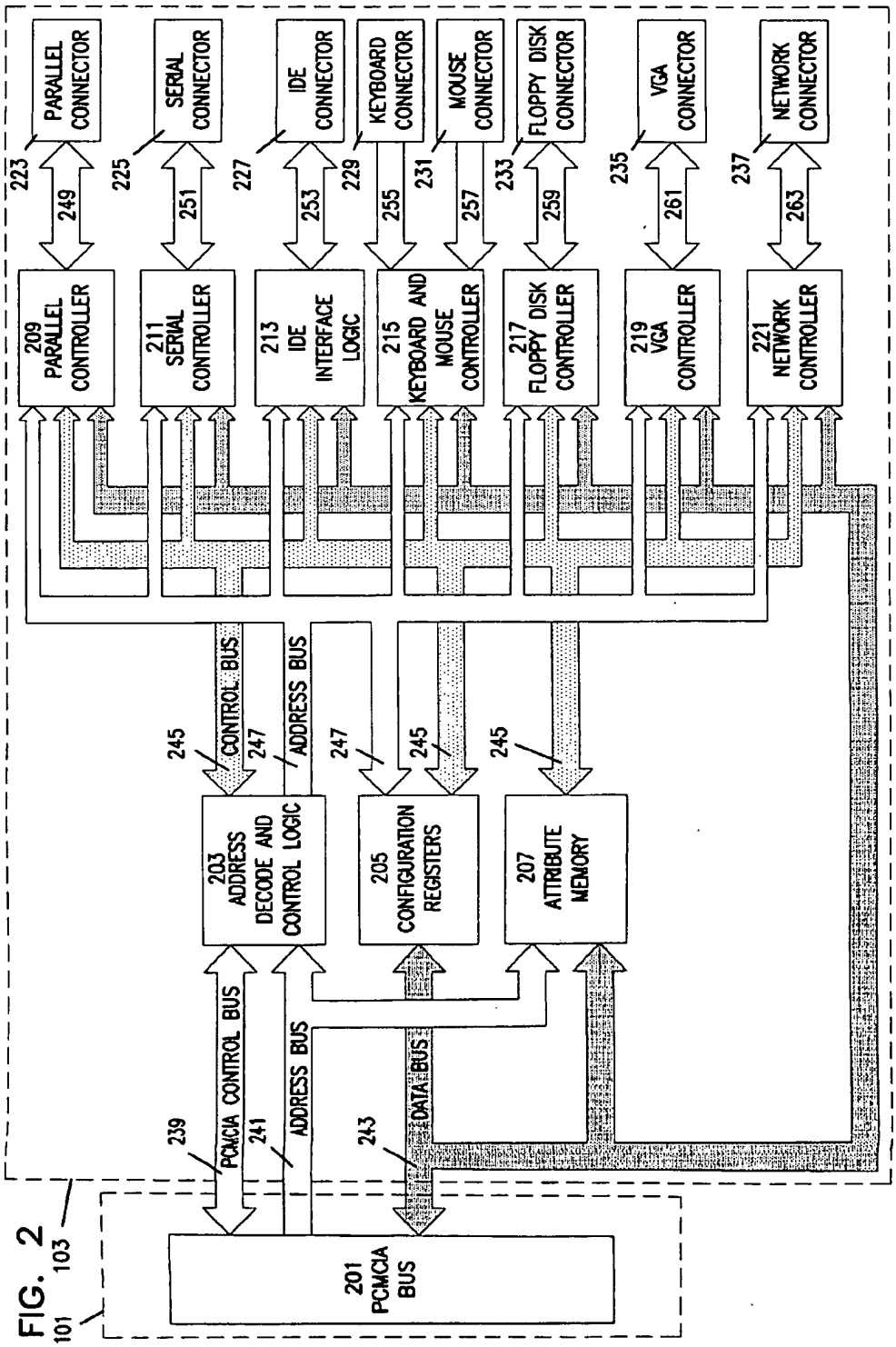

PARALLEL CONNECTOR
SERIAL CONNECTOR
IDE CONNECTOR
KEYBOARD CONNECTOR
MOUSE CONNECTOR
FLOPPY DISK CONNECTOR
VGA CONNECTOR
NETWORK CONNECTOR
223
225
227
229
231
233
235
237
249
251
253
255
257
259
261
263
209 PARALLEL CONTROLLER
211 SERIAL CONTROLLER
213 IDE INTERFACE LOGIC
215 KEYBOARD AND MOUSE CONTROLLER
217 FLOPPY DISK CONTROLLER
219 VGA CONTROLLER
221 NETWORK CONTROLLER
CONTROL BUS
ADDRESS BUS
245
247
203 ADDRESS DECODE AND CONTROL LOGIC
205 CONFIGURATION REGISTERS
207 ATTRIBUTE MEMORY
PCMCIA CONTROL BUS
ADDRESS BUS
DATA BUS
239
241
243
201 PCMCIA BUS
103
101

FIG. 2

FIG. 3

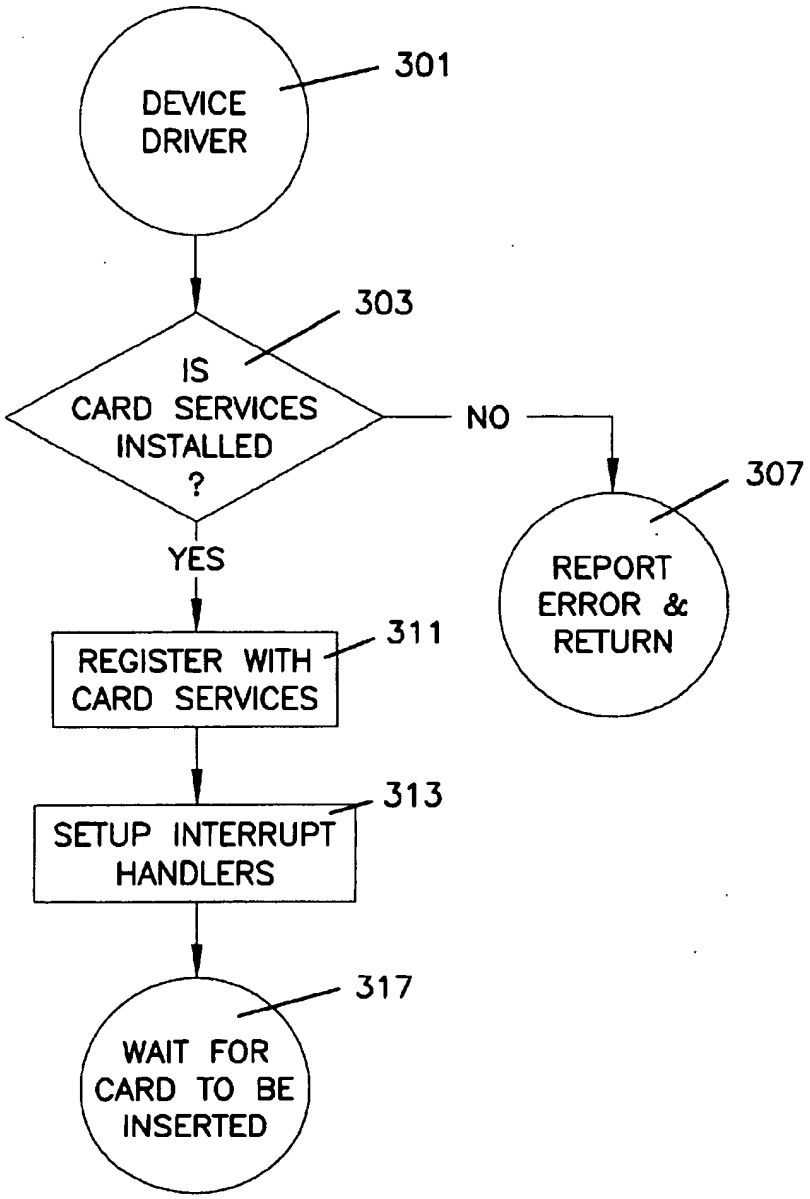

DEVICE DRIVER
301
IS CARD SERVICES INSTALLED ?
303
NO
307
REPORT ERROR & RETURN
YES
REGISTER WITH CARD SERVICES
311
SETUP INTERRUPT HANDLERS
313
WAIT FOR CARD TO BE INSERTED
317

FIG. 4

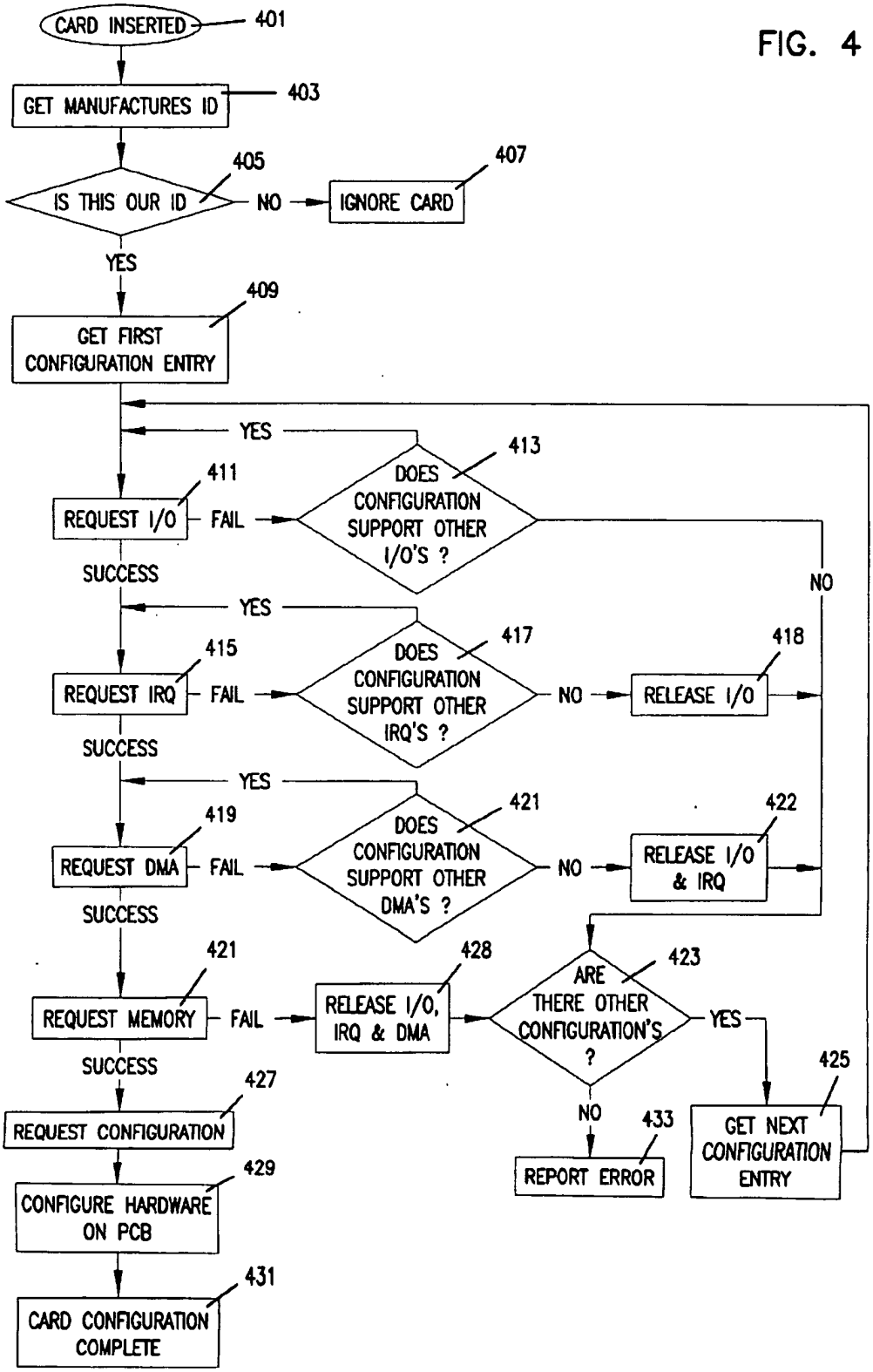

CARD INSERTED
401
GET MANUFACTURES ID
403
IS THIS OUR ID
405
NO
IGNORE CARD
407
YES
GET FIRST CONFIGURATION ENTRY
409
YES
REQUEST I/O
411
FAIL
DOES CONFIGURATION SUPPORT OTHER I/O'S ?
413
NO
SUCCESS
YES
REQUEST IRQ
415
FAIL
DOES CONFIGURATION SUPPORT OTHER IRQ'S ?
417
NO
RELEASE I/O
418
SUCCESS
YES
REQUEST DMA
419
FAIL
DOES CONFIGURATION SUPPORT OTHER DMA'S ?
421
NO
RELEASE I/O & IRQ
422
SUCCESS
REQUEST MEMORY
421
FAIL
RELEASE I/O, IRQ & DMA
428
ARE THERE OTHER CONFIGURATION'S ?
423
YES
NO
REPORT ERROR
433
GET NEXT CONFIGURATION ENTRY
425
SUCCESS
REQUEST CONFIGURATION
427
CONFIGURE HARDWARE ON PCB
429
CARD CONFIGURATION COMPLETE
431

FIG. 5

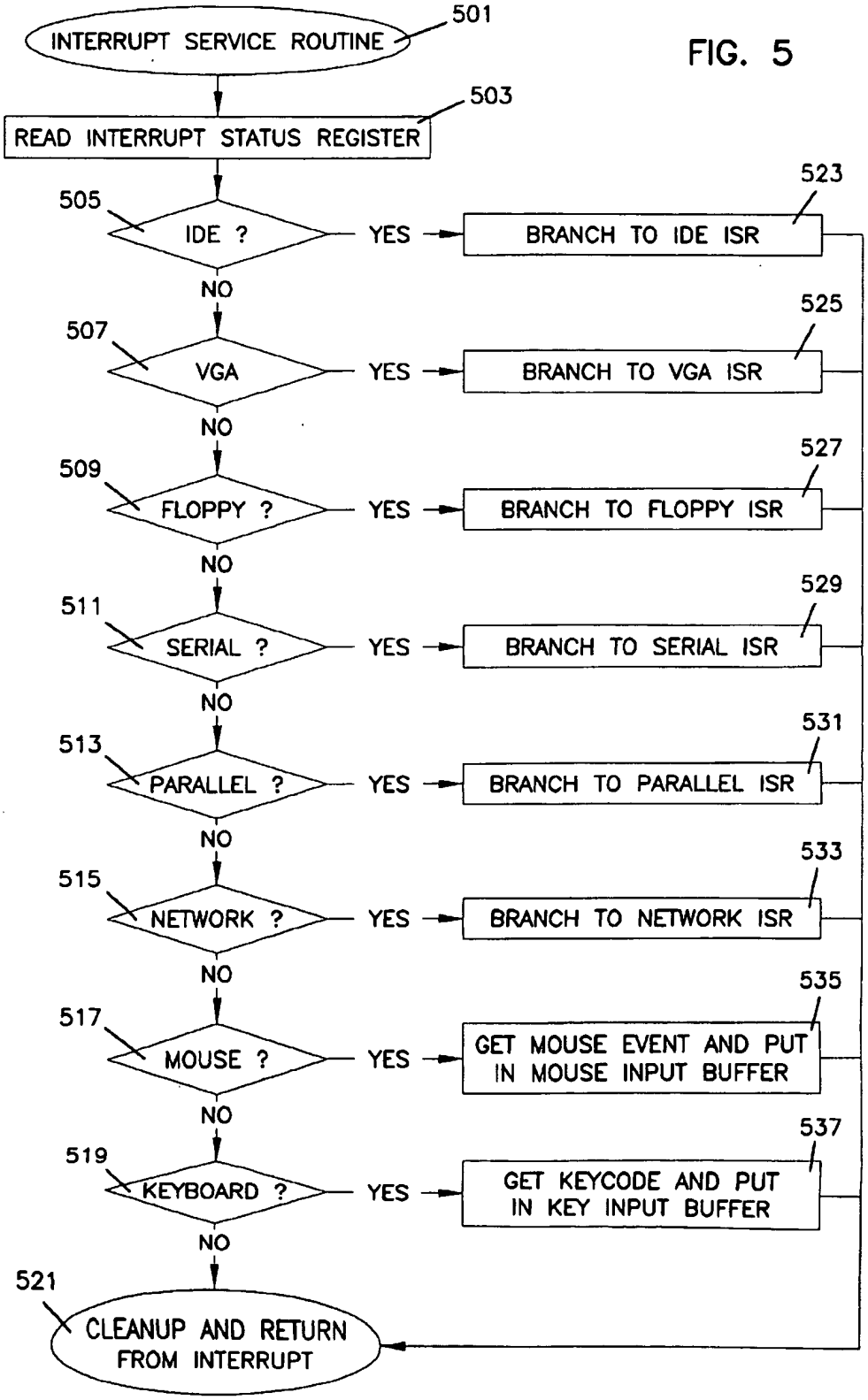

INTERRUPT SERVICE ROUTINE
501
READ INTERRUPT STATUS REGISTER
503
505
IDE ?
YES
BRANCH TO IDE ISR
523
NO
507
VGA
YES
BRANCH TO VGA ISR
525
NO
509
FLOPPY ?
YES
BRANCH TO FLOPPY ISR
527
NO
511
SERIAL ?
YES
BRANCH TO SERIAL ISR
529
NO
513
PARALLEL ?
YES
BRANCH TO PARALLEL ISR
531
NO
515
NETWORK ?
YES
BRANCH TO NETWORK ISR
533
NO
517
MOUSE ?
YES
GET MOUSE EVENT AND PUT IN MOUSE INPUT BUFFER
535
NO
519
KEYBOARD ?
YES
GET KEYCODE AND PUT IN KEY INPUT BUFFER
537
NO
521
CLEANUP AND RETURN FROM INTERRUPT

FIG. 6

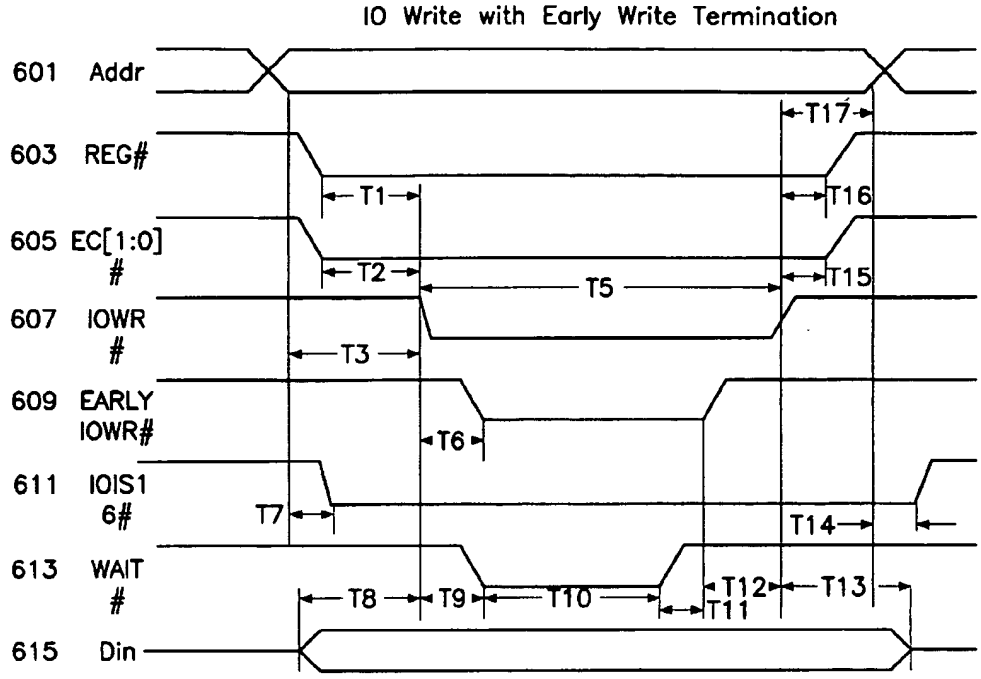
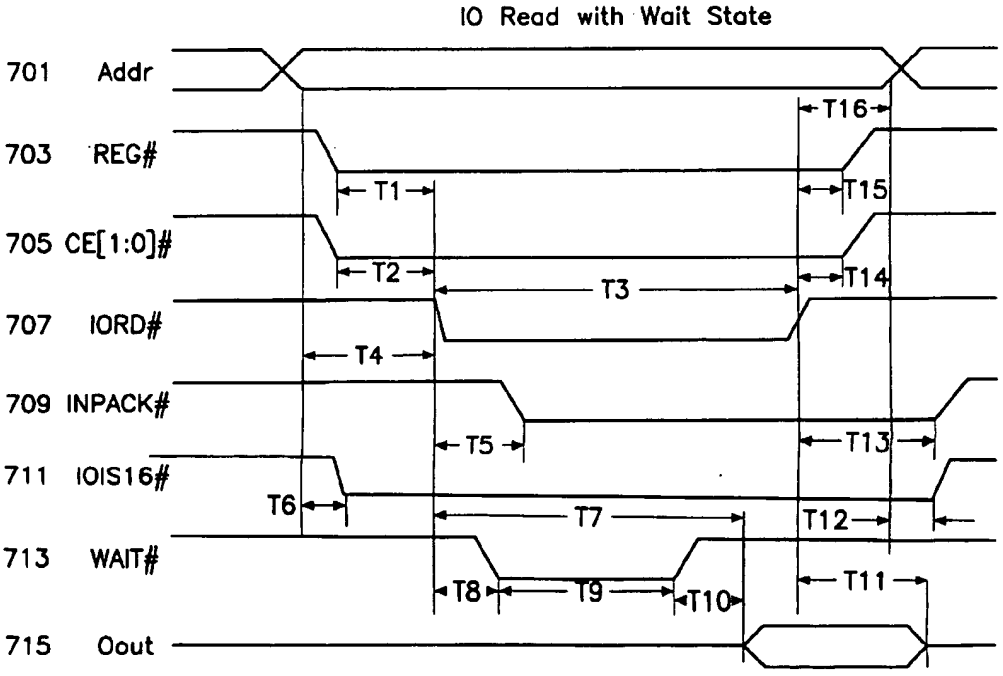

IO Write with Early Write Termination
601 Addr
603 REG#
605 EC[1:0] #
607 IOWR #
609 EARLY IOWR#
611 IOIS1 6#
613 WAIT #
615 Din
T1
T2
T3
T5
T6
T7
T8
T9
T10
T11
T12
T13
T14
T15
T16
T17

FIG. 7


IO Read with Wait State
701 Addr
703 REG#
705 CE[1:0]#
707 IORD#
709 INPACK#
711 IOIS16#
713 WAIT#
715 Oout
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
T11
T12
T13
T14
T15
T16

UNIVERSAL DOCKING STATION

This application claims benefit of Provisional Application 60/017,725 filed May 16, 1996.

FIELD OF THE INVENTION

The present invention relates generally to computers nd more particularly to docking stations used to interface portable computer to multiple peripheral devices.

BACKGROUND OF THE INVENTION

Though popular, portable computers, such as notebook, laptop or palmtop computers have several shortcomings when compared to conventional desktop computers. They typically include a keyboard that is smaller and more difficult to use than a conventional keyboard, and a smaller, lower resolution screen than a conventional desktop monitor. In addition, portable computers rarely include such peripherals as CD ROMS, tape backups, secondary hard drives, modems, and network connectors.

A portable computer user seeking these features has limited options. The user can purchase separate portable and desk top computers. However, given that the user can only use one computer at a time, this option is costly and requires frequent data transfers between the two computers. A second option is to plug the various peripherals into their designated ports on the portable computer. Unfortunately, most portable computers do not have connectors for many of the peripherals desired. Moreover, separately connecting and unconnecting the various peripherals is time consuming and burdensome.

A third option for the portable computer user seeking to expand the capabilities of their portable computer is to purchase a docking station or expansion base into which the particular portable computer may easily be docked during desktop use. Thus, only one computer is necessary, and data transfer is not required. The docking station typically sits on the user's desk and provides connections to various peripheral devices, such as full-size keyboards and monitors, modems, network connectors, etc. Once the portable computer is docked in the docking station, the portable computer has access to all of the various peripherals attached to the docking station. When travel is necessary, the user can simply remove the portable computer from the docking station and carry it with him on the road.

A major shortcoming of current docking stations is their reliance on proprietary connectors to connect the portable computer to the docking station. Consequently, one must purchase the docking station that corresponds to the make and model of the portable computer they currently own, and is most likely precluded from using docking stations manufactured by different companies. This severely limits the usefulness of the docking station concept because a portable computer user is confined to a small number of stations into which he can dock his computer. A strong need exists for a universal docking station that provides the added advantages and capabilities of a desktop computer, particularly a full-size keyboard and monitor, but does not require a proprietary connection to the portable computer.

SUMMARY OF THE INVENTION

A universal docking station for connecting a portable computer to a plurality of peripheral devices is provided, wherein at least one of the peripheral devices is a user input device, at least one of the user input devices is capable of receiving a write command, and the portable computer is capable of writing data. The docking station comprises a PCMCIA interface adapted to be coupled to the portable computer, peripheral control units adapted to be coupled to the plurality of peripheral devices, and a docking station control unit coupled to the PCMCIA interface and the peripheral control units, wherein the docking station control unit selectively transfers data between the PCMCIA interface and one of the peripheral control units, and wherein the docking station control unit extends the period of time data written by the computer remains stable on the PCMCIA interface.

A computer system is also provided comprising a portable computer having a PCMCIA interface, the portable computer capable of generating output signals for designated peripheral devices, a control unit having a PCMCIA interface for interfacing with the portable computer PCMCIA interface, a plurality of peripheral devices coupled to the control unit, with at least one of the peripheral devices being a user input device capable of generating input signals, and at least one of the peripheral devices being a display device, the control unit comprising means for converting the output signals from the portable computer into a form compatible with the designated peripheral device, means for routing the converted signal to the designated peripheral device, means for converting the input signal into a form compatible with the portable computer, and means for prioritizing access to the PCMCIA bus by the input signals. In one embodiment, the means for converting the output signal into a form compatible with the designated peripheral comprises extending the period of time the output signal remains stable.

A method for interfacing a standard peripheral device to a computer via a PCMCIA bus is also provided. The method comprises the steps of driving data to be written to the peripheral device on the PCMCIA bus, extending the time the data is available to the peripheral device on the PCMCIA bus for a period sufficient to satisfy the timing requirements of the peripheral device, and writing the data to the peripheral device. In one embodiment, the extending step is accomplished by activating the write signal a period before a standard write signal.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of this invention reference should now be made to the embodiment(s) illustrated in greater detail in the accompanying drawings and described below by way of examples of the invention.

FIG. 1 is a block diagram showing a docking station in accordance with one embodiment of the invention connected to various peripheral devices and connected to a computer via a standard universal interface.

FIG. 2 is a more detailed block diagram of the docking station represented in FIG. 1.

FIG. 3 is a flow chart showing a method of initializing a device driver into an operating system.

FIG. 4 is a flow chart of steps that occur in handling insertion of a PCMCIA card into a PCMCIA port of a computer.

FIG. 5 is a flow chart of steps that occur when the docking station shown in FIG. 1 causes an interrupt on the computer requesting service for one of the peripheral devices.

FIG. 6 is a timing diagram showing the write timing used to interface multiple peripheral devices to the PCMCIA port.

FIG. 7 is a timing diagram showing the read timing used to interface multiple peripherals to the PCMCIA port.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The universal docking station or expansion base of the present invention allows a portable computer user to interface a portable computer to several different peripheral devices, such as CD-ROMs, Hard Disk Drives, Floppy Disk Drives, Tape Backups, standard size keyboards and mice, standard size VGA/super VGA monitors, networks, and other peripherals that typically utilize serial and/or parallel ports.

Rather than using proprietary connectors to connect the computer to the docking station, the present invention accomplishes this task using a standard universal interface. One such interface is the Personal Computer Memory Card International Association ("PCMCIA") slot, port or socket provided on most portable computers. To interface with the PCMCIA slot, the docking station uses a PCMCIA card ("PC Card") as the connection between the portable computer and the docking station. With this arrangement, portable computer users can connect their portable computers to multiple peripherals with one PCMCIA card. Because PCMCIA ports on portable computers are almost identical physically and electrically from computer to computer, the present docking station will work with almost any portable computer having such a port. With this arrangement, the docking station of the present invention can be positioned not only at a user's office and home, but at airports, libraries, business associates" offices, and virtually anywhere else computers are used. Portable computer users would no longer be limited by the make and model of computer they carry.

Background on 16-Bit PCMCIA cards can be found in Mindshare, Inc. & Don Anderson, *PCMCIA System Architecture 16 Bit PC Cards, Second Edition* (Addison-Wesley Publishing Company 1995), which is hereby incorporated by reference. Background on 32-bit PCMCIA standard known as "card bus" can be found in Mind Share, Inc., *Card Bus System Architecture* (Addison-Wesley Publishing Company 1995), which is hereby incorporated by reference.

Referring to the drawings, particularly FIG. **1**, a portable computer **101** having a PCMCIA port **102** is connected to a PCMCIA card **105** which interfaces to a docking station **103**. The PCMCIA card **105** completes the interface between the portable computer **101** and the docking station **103**. Several peripheral devices **106** are coupled to the docking station **103**. Possible peripheral devices include: keyboard **107**, joy stick **109**, mouse **111**, modem **113**, network interface **115**, hard disk **117**, floppy disk **119**, CD ROM **121**, tape backup **123**, printer **125**, and monitor **127**.

In one embodiment, the docking station **103** contains two standard expansion slots **129** and **131** configured and arranged to receive any combination of two of the following standard internal peripheral devices: hard disk **117**, floppy drive **119**, CD ROM **121** and tape backup **123**. Other embodiments of the present invention may provide for additional expansion slots for receiving additional internal peripheral devices.

As indicated above, the portable computer **101** contains at least one PCMCIA slot. The portable computer **101** also includes a central processing unit (CPU) coupled to a Read Only Memory (ROM) and Random Access Memory (RAM). The computer communicates with the PCMCIA slot and other internal and external components through an internal or input/output (I/O) bus. A controller or Host Bus Adaptor (HBA) links signals coming from the PCMCIA slot (or from a PCMCIA card installed in the slot) to the I/O bus of the portable computer. The computer may also include one or more data storage devices, such as a hard disk drive, a floppy disk drive, and CD-ROM drive. In one embodiment, software used in connection with the present invention may be stored and distributed on a CD-ROM, which may be inserted into and read by the CD-ROM drive. The computer is also coupled to a display, and a user input device such as a mouse or keyboard.

A memory window can be created in the portable computer's address space into which memory and configuration registers of the PCMCIA card can be individually mapped. This memory window can be set up by a device driver of the PCMCIA card, and typically remains the same size and keeps the same memory location on the portable computer.

Referring to FIG. **2**, the docking station **103** interfaces to the computer **101** via a PCMCIA bus **201** on the computer **101**. The computer **101** includes a PCMCIA slot or socket connected to the PCMCIA bus **201**, into which a PCMCIA card can be inserted to connect the PCMCIA card to the PCMCIA bus **201**. As used herein, a PCMCIA card refers, generically, to a standardized interface between a peripheral device and an internal bus of a computer. Typically, the PCMCIA card will be of a standard length and width, and will have a thickness determined by the card type (e.g., Type I =3.3 mm thick; Type II =5.0 mm thick; Type III =10.5 mm thick). The PCMCIA card is configured to fit into a PCMCIA slot or socket. When inserted into the PCMCIA slot or socket, the PCMCIA card can be connected to a wide variety of host buses, typically via host bus adapters designed for a particular bus interface. The PCMCIA bus **201** is an expansion of the computer's internal bus, and allows devices connected to the PCMCIA port to be accessed by the computer as if they were inside the computer. The PCMCIA physical interface allows for devices to be inserted and removed at any time from the computer.

The PCMCIA bus **201** operatively couples a PCMCIA control bus **239**, an address bus **241** and a data bus **243** to core logic **202** of the docking station **103** to the computer **101**. The core logic **202** includes address decode and control logic **203**, configuration registers **205**, and attribute memory **207**.

In one exemplary embodiment, the address decode and control logic **203** is implemented using field programmable gate arrays (FPGA). Alternatively, the address decode and control logic **203** could be implemented using a program array logic (PAL) or other similar programmable devices or custom integrated circuits (IC) so long as the particular implementation can handle the timing requirements of the PCMCIA bus **201** as well as all of the peripheral devices **106** connected to the docking station **103**.

In the exemplary embodiment, the configuration registers **205** contain five registers. The first configuration register is a standard PCMCIA register needed for all PCMCIA devices, commonly referred to as the configuration option register. This register contains 8 bits to enable the PCMCIA card to behave as an I/O card and also has a bit that resets the card to a known state. The second configuration register is a 16 bit register that contains the I/O address for the configuration registers of the different peripheral devices attached to the docking station **103**. The third register is an interrupt flag which is an 8 bit register that contains a bit for each device to interrupt the computer for services. The fourth register is a 16 bit register that is loaded with the address for the IOIS16 signal used by the IDE interface. The IOIS16 is a signal used to inform the computer that a device desires to carry out a **16** bit transfer as opposed to an 8 bit

transfer to the computer. The last register is a keyboard configuration register which is a 16 bit register that is loaded with the I/O address that the keyboard controller needs to be mapped to in the system.

In the above described exemplary embodiment, the configuration registers **205** are implemented using a field programmable gate array (FPGA). The configuration registers **205** could also be implemented using random access memory (RAM) or a programmable array logic (PAL) device. Different size registers may also be sued as dictated by the actual implementation.

The attribute memory **207** is implemented in the exemplary embodiment with an electronically erasable programmable read only memory (EEPROM) or other suitable standard nonvolatile memory. In one embodiment, only about 1024 bytes of memory are required to store all the values needed for the docking station **103**.

Peripheral devices **106** can be connected to the docking station **103** through appropriate connectors (**223-237**). Once connected, the peripheral devices **106** interface through the address decode and control logic **203**.

A parallel connector is connected through connection **249** to a parallel controller **209**. The parallel controller **209** is connected to the address decode and control logic **203** via control bus **245** and address bus **247**. Parallel controller **209** is also connected to configuration registers **205** with control bus **245** and address bus **247**. A data bus **243** is linked to, and can provide data to, the configuration registers **205**, the attribute memory **207**, and the parallel controller **209**. The parallel controller **209** may be implemented, for example, using a standard **8255** compatible parallel controller used on IBM XT/AT compatible computers. This supports the optional PS/2 bidirectional parallel port (SPP), the Enhanced Parallel Port (EPP) and the Extended Capabilities Port (ECP) modes. This interface is useful for connecting printers, removable media high density storage devices and scanners to the docking station **103**. A Standard Microsystems Corporation (SMC) FDC37C93X Plug and Play Compatible Ultra I/O Controller includes a parallel port and can be used for this purpose.

A serial connector **225** is connected via connection **251** to a serial controller **211**. The serial controller **211** is connected to the address decode and control logic **203**, configuration registers **205** by way of control bus **245**, address bus **247** and data bus **243**, as indicated in FIG. **2**.

The serial controller **211** may be a **NS16C550** compatible serial controller or other serial controller that can handle high speed communication (i.e., communication above 460K Baud), and has a built in FIFO for handling data received by the serial port at a rate faster than can be sent through the interface to the computer. The serial controller may be a standard 16C550 compatible Universal Asynchronous Receiver/Transmitter (UART), for example, with a 16 byte FIFO. The UART performs the serial-to-parallel conversion for receiving characters and the parallel-to-serial conversion for transmitting characters. This UART allows for data rates from 50 to 460.8K baud. The character options are programmable for 1 start; 1, 1.5 or 2 stop bits; even, odd, sticky or no parity; and prioritized interrupts. The UART contains a programmable baud rate generator that is capable of dividing the input clock or crystal by a number from 1 to 65535. The UART is also capable of supporting Musical Instrument Digital Interface (MIDI) data rate. An SMC FDC37C93X Plug and Play Compatible Ultra I/O Controller can be used for this purpose. Other serial controllers may also be used so long as they support a communications speed of 460K baud and contain a FIFO for handling data overflow.

An Integrated Drive Electronic (IDE) connector **227** is connected via connection **253** to IDE interface logic **213**. The IDE enables hard disk drives with embedded controllers to be interfaced to the host processor. The IDE interface performs the address decoding for the IDE device. This interface also supports devices such as CD-ROM drives and newer high density removable storage devices. The IDE **213** includes an address decoder for the specific drive or mass storage device to be interfaced to, and interrupt circuitry to allow the device to request service from the computer. The interrupt source goes back through the control bus **245** through the address decode and control logic **203** back into the PCMCIA bus **201**. In an exemplary embodiment, the SMC FDC37C93X Plug and Play Compatible Ultra I/O Controller provides the IDE interface.

A keyboard connector **229** is connected via connection **255** to a keyboard and mouse controller **215**. A mouse connector **231** is also connected via connection **257** to the keyboard and mouse controller **215**. The keyboard and mouse controller **215** interfaces through the control bus **245**, address bus **247** and data bus **243**. The keyboard and mouse controller **215** should contain means for communicating to a keyboard and a mouse and means for interfacing with a computer. The keyboard and mouse controller **215** may be implemented using a universal keyboard control with a standard Intel 8042 micro controller CPU core. The SMC FDC37C93X Plug and Play Compatible Ultra I/O Controller, for example, provides the keyboard and mouse controller **215**. Other standard keyboard and mouse controllers may also be used.

A floppy disk connector **233** is connected to the floppy disk controller (FDC) **217** via connection **259**. The FDC **217** is connected to the control bus **245**, address bus **247** and data bus **243**. An IBM compatible FDC can be used, and preferably one with a CMOS **755** floppy disk controller that supports a 2.88 megabyte super floppy drive. This FDC **217** can handle up to two floppy disk drives or tape backups. The FDC integrates the functions of the Formatter/ Controller, Digital Separator. Write Precompensation and Data Rate Selection logic for IBM XT/AT compatible FDC are also provided. The true CMOS **765B** core guarantees 100% IBM PC XT/AT computability in addition to providing data overflow and underflow protection. In an exemplary embodiment, the SMC FDC37C93X Plug and Play Compatible Ultra I/O Controller provides the FDC.

A VGA connector **235** is connected to a VGA controller **219** via connection **261**. The VGA connector **235** interfaces through the control bus **245**, address bus **247** and data bus **243**. The VGA controller **261** can have some VGA memory integrated into it, support up to 1024 by 756 pixels, and be compatible with a super VGA monitor.

A network connector **237** is connected to a network controller **221** through connection **263**. The network controller **221** interfaces through the control bus **245**, address bus **247** and data bus **243**. Between a 10 megabyte and a 100 megabyte controller can be supported. In the exemplary embodiment, the network controller **221** is an Ethernet controller.

As indicated, the keyboard and mouse controller **215**, FDC **217**, parallel controller **209**, serial controller **211**, and IDE interface **213** can be implemented using a Standard Microsystems Corporation (SMC) FDC37C93X Plug and Play Compatible Ultra I/O Controller. This device incorporates a keyboard interface, SMC's true CMOS 765B floppy disk controller, advance digital separator, 16 byte data FIFO, 16C550 compatible UARTs, a Multi-Mode parallel port and

an IDE interface. The FDC37C93X also provides support for the ISA Plug-and-Play Standard (Version 1.0a) and provides for the recommended functionality to support Windows '95.

Most PCMCIA socket controllers designed into most computers have a limited I/O window size. Typically, only two I/O windows are permitted. This will allow for one or two functions to have I/O ports. With more than two functions on the docking station it is necessary to combine all of the I/O ports into two contiguous pieces of I/O memory.

Almost all of the functions on the SMC chip can be relocated in I/O address space via configuration registers. The only function that is fixed is the keyboard and mouse controller. To accommodate this a Xilinx FPGA is built into the hardware of the docking station. The Xilinx device includes logic to match an address and output the appropriate address to the SMC chip.

FIG. **3** shows a flow chart depicting steps performed in inserting the docking station device driver **301** into the RAM of the portable computer **101**. Processing begins with decision step **303** which detects whether card services (a piece of software that is used to interface with the PCMCIA port at a high level) is installed on the computer; if not, an error is flagged at step **307**, processing stops, and the device driver is not installed into the operating system. If card services is installed, processing continues with step **311** wherein the device driver registers with card services and sets up a call back handler, thereby allowing card services to inform the docking station device driver of events that happen in the PCMCIA port such as a PCMCIA card being inserted or removed. After the device driver is registered with card services, processing continues with step **313** which polls for interrupt vectors and loads the interrupt vectors into memory to allow the device driver to handle different functions of the docking station. Once in memory, the device driver stays in memory waiting for one of the callback events from card services to tell it that the docking station PCMCIA card has been inserted into the computer's PCMCIA port as shown in item **317**.

The PCMCIA standard specifies that all PCMCIA devices must behave as a memory device until configured by the host computer. After configuration, the PCMCIA device must convert some of the interface pins to the PCMCIA bus to support the I/O interface. This is accomplished in one embodiment of the universal docking station **103** by using a Xilinx FPGA to control the functions of the flexible interface pins. All PCMCIA devices must also have memory on board that identifies the device's functions and capabilities. This is accomplished in the exemplary embodiment of the present invention by using a 2K EEPROM that is accessible by the computer at any time. The EEPROM also allows for software to write new information to it when upgrades or modifications are necessary.

FIG. **4** depicts the steps performed during card configuration. Processing begins at step **401** when a PCMCIA card is inserted into the PCMCIA port on the computer, and card services informs the device driver through the callback handler that a card has been inserted. Next, in step **403**, the device driver asks card services for the manufacture ID of the card that was inserted. If the card inserted is the docking station PCMCIA card, the manufacturer ID is stored in attribute memory **207**. In decision step **405**, the manufacturer ID of the card inserted is compared to the ID designated for the docking station; if no match is found, the system ignores the card as shown in step **407** and continues waiting for another card to be installed. If the manufacturer ID is correct, processing continues with step **409** which gets the first configuration entry that is stored in the attribute memory on the card.

Next, the device driver tries to configure the I/O port and different interfaces needed for using the docking station card in the system given the particular configuration entry. The first step is the I/O port step **411** which is requested from card services and card services either allows the docking station card to have the I/O port or not. If not, the system will determine in decision step **413** whether the configuration allows other I/O ports to be used. If there are more I/O ports in this particular configuration, the system will return to step **411** to try requesting the next I/O port. This routine will continue until an I/O port is successfully requested or all I/O ports of this particular configuration entry are exhausted. If there are no more I/O ports in the particular configuration, decision step **423** determines whether there are other configurations available. If so, step **425** gets the next configuration entry and processing continues at step **411**. If no other configurations exist, an error is reported at step **433**.

If an I/O port is successfully requested in step **411**, the processing will continue with step **415** which requests an interrupt. A similar process will happen with respect to the interrupt as with the I/O port; if the system does not get an interrupt, decision step **417** checks the configuration to see if another interrupt is allowed. If there is a possibility of another interrupt, it will go back and try to request the next interrupt. This process will continue until it either runs out of possible interrupts to try or it successfully allocates an interrupt for use with the docking station card. If it runs out of interrupts to try, step **418** releases the I/O port previously requested in step **411**, and decision step **423** determines whether other configurations exist. If not, step **433** reports an error. If so, step **425** gets the next configuration entry and processing starts over at step **411**.

If an interrupt is successfully requested in step **415**, step **419** requests a direct memory address (DMA) channel. If the request fails, decision step **421** determines whether the configuration supports other DMA'S. If so, step **419** will request that DMA, and this process will continue until it either runs out of possible DMAs or it successfully allocates a DMA channel. If decision step **421** determines that no more DMAs exist for the particular configuration, step **422** releases the I/O port and the interrupt, and moves to decision step **423** to determine if other configurations are available. If so, step **425** gets the next configuration and processing continues at step **411**. If not, step **433** reports an error.

If a DMA channel is successfully requested in step **419**, step **421** requests a memory block. If step **421** fails, step **428** releases the I/O port, the IRQ, and the DMA channel. Next, decision step **423** determines whether another configuration is available. If not, step **433** reports an error. If so, step **425** gets the next configuration entry and processing continues from step **411**.

Upon successfully requesting the I/O, the IRQ, DMA and memory, step **427** requests configuration for the docking station, step **429** configures the hardware on the docking station board, and step **431** indicates that card configuration is complete.

FIG. **5** shows a flowchart of the interrupt service routine. The interrupt service routine comprises a series of steps performed when one of the peripheral devices **106** connected to the docking station **103** requests service from the computer **101** by setting its interrupt flag in the interrupt flag register of the control logic **203**.

The PCMCIA specification and available controller hardware support only one interrupt per card. This presents a problem when more than one function on a card needs to interrupt the host computer for servicing. The PCMCIA 1995 specification solves this issue by having the card services (high level interface to PCMCIA devices for applications) handle the interrupts from a card. When a card is configured the card services starts with the first function and assigns it an interrupt if needed. Card services then stores the vector for this interrupt in a table and places its own interrupt vector in the appropriate memory location. When card services configures the next function on the card it stores that function's interrupt vector in the table next to the first function's interrupt vector. This continues until all functions are configured.

When an interrupt occurs, card services' interrupt handler is called from the vector in memory. Card services will then interrogate the PCMCIA cards interrupt status register to identify the source of the interrupt. Card services will then call the appropriate service routine for the function by calling the function pointed to by the vector in the stored table.

Step **503** reads an interrupt status register on the docking station **103**. In a preferred embodiment, the interrupt status register will be in the configuration register **205** shown in FIG. **2**, and have one bit in it for each function of the docking station **103**. The interrupt service routine will then look at these bits and decide whether or not one of the devices is in need of service. If any of the devices connected to the expansion box set their interrupt flag in the interrupt flag register, the control logic **203** will then send the interrupt flag to the PCMCIA bus which will cause a hardware interface on the particular interrupt line that has been configured in accordance with FIG. **4**.

After the interrupt service handler has read the interrupt status register, it decides which interrupts to handle in a prioritized fashion starting with the IDE interface in decision step **505**, and proceeding downward in priority as shown in FIG. **5**, until it reaches the keyboard which is lowest in priority. If decision step **505** determines that the flag is set for the IDE, then step **523** branches to the IDE interrupt service routine. Upon completion of this interrupt service routine, step **521** cleans up any registers that may have information pushed on to their stacks and returns from the interrupt. If the flag is not set for the IDE, decision step **507** determines whether the interrupt flag is set for the VGA. If so, step **525** branches to the VGA interrupt service routine.

If decision step **509** determines that the flag is set for the floppy disk controller, step **527** branches to the floppy interrupt service routine. If decision step **511** determines that the flag is set for the serial port, step **529** branches to the serial interrupt service routine. If decision step **513** determines that the flag is set to the parallel port, step **531** branches to the parallel port interrupt service routine. If decision step **515** determines that the flag is set to a network, step **533** branches to the network interrupt service routine. If decision step **517** determines that the flag is set to the mouse, step **535** gets the mouse event from the keyboard controller and puts it in the mouse input buffer. If decision step **519** determines that the flag was set by the keyboard, step **537** gets the keystroke from the input buffer of the keyboard controller and puts it into the BIOS input buffer of the operating system on the computer.

Although priority may be altered from that shown in FIG. **5**, the priority shown has certain advantages over other possible schemes because of the relative speed of each device and the capability of the devices to hold data before being manipulated. Those devices that have room to store data for longer periods of time (e.g., keyboard) will be lower in priority than other devices which will throw away this information in a short period of time if it does not get taken out of the buffer.

An example of the serial controller **211** taking control of the PCMCIA bus **201** follows. The serial controller **211** would be configured to receive information from the serial connector **225** and would then send this data through the connections of **251** into the serial controller **211**. The serial controller **211** would then request servicing to the computer **101** by setting the interrupt flag through the control bus **245** to the configuration registers **205**. The configuration registers **205** would then set the interrupt flag through the control bus **245** to the address, decode and control logic **203** and the decode and control logic would send the signal through **239** to the PCMCIA bus **201**.

FIG. **6** shows a timing diagram for I/O write timing and illustrates a solution to the type of problems encountered when interfacing standard peripheral devices **106** to the PCMCIA port. Because PCMCIA was originally designed for memory devices, the specifications for interfacing with I/O devices are more constrained than other specifications for interfacing with I/O devices. Nonetheless, in order to interface with I/O devices through the PCMCIA port, one must somehow meet these constraints. This problem has been addressed conventionally by designing the I/O device with the PCMCIA constraints in mind, and, therefore, by designing the constraints right into the device. In contrast, the present invention allows the use of standard off-the-shelf devices that would normally be connected to the ISA (Industry Standard Architecture) bus of a computer, and use them on the PCMCIA bus. In other words, the PCMCIA interface is used in the present invention in a manner not contemplated by its design, and yet is also being used to interface off-the-shelf peripherals. The present invention addresses these competing interests.

Most IDE devices are designed to conform to the ANSI AT Attachment Interface for Disk Drives specification (ANSI X3.221-1994). This specification was designed to interface IDE devices directly to an Industry Standard Architecture (ISA) bus. This bus differs in some ways from the PCMCIA bus. In order to allow a standard IDE device to be connected to the docking station, the timing of some of the main control signals is modified since the timing constraints of a standard IDE device do not match that of the PC bus.

For example, the specification for PCMCIA does not require that the data bus be held active for any amount of time after a write signal goes high, but I/O devices need the data bus to be stable for a period of time after the I/O write signal goes inactive. The present invention overcomes this problem by extending the period of time the data bus is allowed to remain stable. In an exemplary embodiment, the period of time the data bus is allowed to remain stable is extended by activating early write signals.

To accomplish this task a Xilinx FPGA was designed into the hardware of the docking station. This device allows for a large amount of flexible glue logic to be utilized without taking up a large amount of printed circuit board real-estate and for a reasonable cost.

By way of example, if the computer **101** writes a byte of information to a peripheral device **106**, e.g., the serial controller, the computer **101** first sets up the address bus signal **601**. When the address bus signal **601** settles out and

is determined to be the correct address as in the address and decode logic **203** in FIG. **2**, the computer **101** waits for a short period of time and then either activates the IOIS16 signal **611** (for a 16 bit transfer) or does not activate the IOIS16 line **611** (for an 8 bit transfer). The IOIS16 signal **611** is used to inform the host computer that the IDE device would like to transfer 16 bits of data instead of 8. This signal is output from an IDE device and is active within 90ns from the activation of a valid address. The PCMCIA spec requires this signal to be active within 35ns of a valid address.

To overcome this problem the Xilinx FPGA on the docking station main printed circuit board (PCB) has a logic circuit in it to match the valid address and assert the IOIS16 line within the PCMCIA timing constraint.

Then the computer **101** activates the register signal **603** in the PCMCIA port. At about the same time, the computer activates CE[1:0] **605**. CE[0] would be activated for an 8 bit transfer and both CE[1:0] would be activated for a 16 bit transfer. About a third period of time after that, the computer would activate the I/O write signal **607** letting the peripheral device that is addressed activate the address line **601**, telling the device to set up the address line **601** of the data bus for the value that is going to be written to it. The Din signal **615** represents the data being sent by the computer. It settles out after a period of time. The peripheral device also sets the wait signal **613** to tell the computer to extend this write period for a longer period of time than a standard write. The Wait signal is another signal that could not meet the time constraints of the PCMCIA interface. This signal is asserted by an I/O device to inform the host computer that it needs more time to accomplish a task. The timing specs for PCMCIA and IDE both require this signal to be active within 35ns. However, because all of the other timing critical signals must run through the Xilinx and other hardware on the PCB, this timing was not possible to meet. Therefore the Xilinx FPGA was used to activate an early wait signal. This signal delays the host computer by 100 to 500 ns to allow the other signals to settle.

At that point, the hardware on the docking station sets the early I/O write signal **609** so that the peripheral device, in this case the serial port, would see the actual write signal happen and on the other edge of that write signal (i.e., the rising edge) the device would then write the data on the data bus **615** into the particular register that is mapped out by the address bus **601**. This early I/O write signal is activated a period of time before the actual I/O write signal occurs to allow the device to have valid data on the data bus **615** for a longer period of time. The I/O write signal then goes inactive, the CE[1:0] signal **605** goes inactive, and the read signals go inactive. Finally, the end of the address signal **601** becomes unstable.

FIG. **7** shows a timing diagram showing the read timing used to interface with the PCMCIA port. The address bus **701** is the first signal to become stable and settle on the bus to indicate to the docking station **103** that one of the peripheral devices **106** on the docking station is to be addressed. The REG signal **703** then goes low and CE[1:0] (again, CE[0] would be activated for an 8 bit transfer and both CE[1:0] would be activated for a 16 bit transfer) becomes low. The control logic **203** in FIG. **2** activates the IOIS16 signal if a **16** bit transfer is desired for this particular read cycle. After the IOIS16, REG, and CE[1:0] signals have become low, the I/O read signal **707** would go low. This would indicate to the peripheral device (e.g., the serial port) that the computer wants to read information from it. The INPACK signal **709** would then go low to access a PCMCIA card register and notify the HBA that access belongs to the PCMCIA card. This allows the PCMCIA interface to drive the signal onto the internal bus of the computer. The docking station would then bring the wait signal **713** low to extend the time of the read cycle from the computer. This causes the computer to stretch out the read timing to allow the peripheral device to activate the data output **715** in a reasonable amount of time so that the computer can handle reading this information back and so that data would then become stable on the data bus. The read signal **707** would then be brought back high causing the computer to read in the information on the data bus. On completion of the actual read on the rising edge of the read signal **707**, the CE[1:0] **705** and REG signals **703** would then be brought back high. The address signal **701** would then become unstable or start transitioning the next read value item **701** and a short time after that INPACK **709** and IOIS16 signal would be deactivated.

When the present invention is implemented without a VGA controller, a 16 bit PCMCIA port may be used. However, if a VGA controller is included in the docking station, a 32 bit PCMCIA port called "card bus" is preferable. This is because a 16 bit PCMCIA port runs at around 8 or 10 megahertz which is a fairly slow rate for updating video memory. A 32 bit PCMCIA port, however, runs at about 33 megahertz, which is similar to a PCI (Peripheral Component Interconnect) bus which is internal to computers. Interfacing the docking station to the card bus interface will provide for the wider bandwidth that is needed to handle video signals. The multiplexing and demultiplexing of the data and address lines must be properly adjusted to the PCMCIA standard chosen to ensure that the entire 16 (or 32) bit address and 16 (or 32) bit data are captured.

The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention. Since many embodiments of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.

We claim:

1. A computer system comprising:

a) a portable computer having a PCMCIA interface, said portable computer capable of generating output signals for designated peripheral devices;

b) a control unit having a PCMCIA interface for interfacing with said portable computer PCMCIA interface;

c) a plurality of peripheral devices coupled to said control unit, with at least one of said peripheral devices being a user input device capable of generating input signals. and at least one of said peripheral devices being a display device;

d) said control unit comprising

i) means for converting said output signals from said portable computer into a form compatible with said designated peripheral devices,

ii) means for routing said converted signal to said designated peripheral device,

iii) means for converting said input signal into a form compatible with said portable computer,

iv) means for prioritizing access to said PCMCIA interface by said input signals; and

e) wherein said means for converting said output signal into a form compatible with said designated peripheral devices comprises extending a period of time the output signal remains stable.

2. A docking station for interfacing a plurality of peripherals to a portable computer having a PCMCIA interface, wherein the portable computer generates output signals for

designated peripherals, and the peripheral devices generate input signals for the portable computer, the docking station comprising:

a) a PCMCIA interface configured to communicate input and output signals with the PCMCIA interface of the portable computer;

b) means for converting output signals generated by the portable computer into a form compatible with the designated peripherals;

c) means for routing said converted signals to said designated peripherals;

d) means for converting input signals generated by the peripheral devices into a form compatible with the designated peripherals;

e) means for prioritizing access by the input signals to said PCMCIA interface; and

f) wherein said means for converting output signals generated by the portable computer into a form compatible with the designated peripherals comprises extending the period of time the output signals remain stable.

3. A method for interfacing a standard peripheral device to a computer via a PCMCIA bus, the method comprising the steps of:

a) driving data to be written to the peripheral device on the PCMCIA bus;

b) extending the time the data is available to the peripheral-device on the PCMCIA bus for a period sufficient to satisfy the timing requirements of the peripheral device; and

c) writing the data to the peripheral device.

4. The method of claim 3 wherein the extending step is accomplished by activating the write signal a period before a standard write signal.

5. A universal docking station for connecting a portable computer to a plurality of peripheral devices, wherein at least one of the peripheral devices is a user input device, at least one of the user input devices is capable of receiving a write command, and the portable computer is capable of writing data, the docking station comprising:

a) a PCMCIA interface adapted to be coupled to the portable computer;

b) peripheral control units adapted to be coupled to the plurality of peripheral devices;

c) a docking station control unit coupled to the PCMCIA interface and the peripheral control units;

d) wherein the docking station control unit selectively transfers data between the PCMCIA interface and one of the peripheral control units; and

e) wherein the docking station control unit extends the period of time data written by the computer remains stable on the PCMCIA interface.

* * * * *

US006199122B1

(12) **United States Patent**
Kobayashi

(10) **Patent No.: US 6,199,122 B1**
(45) **Date of Patent: Mar. 6, 2001**

(54) **COMPUTER SYSTEM, EXTERNAL STORAGE, CONVERTER SYSTEM, AND RECORDING MEDIUM FOR CONVERTING A SERIAL COMMAND AND DATA STANDARD TO A PARALLEL ONE**

(75) Inventor: **Toshiya Kobayashi**, Koshigaya (JP)

(73) Assignee: **Tokyo Electron Device Limited**, Yokohama (JP)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/120,326**

(22) Filed: **Jul. 22, 1998**

(30) **Foreign Application Priority Data**

Aug. 1, 1997 (JP) 9-208164

(51) **Int. Cl.**[7] **G06F 13/14**; G06F 13/20

(52) **U.S. Cl.** **710/36**; 714/724; 371/22.1; 371/22.3; 371/22.5; 371/22.6; 709/214; 709/250

(58) **Field of Search** 714/724; 371/22.1, 371/22.3, 22.5, 22.6; 709/250, 214; 710/36

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,276,802	*	1/1994	Yamaguchi et al.	395/164
5,726,768	*	3/1998	Ishikawa et al.	358/442
5,768,147	*	6/1998	Young	364/492
6,055,656	*	4/2000	Wilson, Jr. et al.	714/724
6,070,196	*	5/2000	Mullen, Jr.	709/250

FOREIGN PATENT DOCUMENTS

59-20086		2/1984	(JP) .	
402109426A	*	4/1990	(JP)	H03M/11/04
7-334316		12/1995	(JP) .	
08166919A	*	6/1996	(JP)	G06F/13/12
8-227359		9/1996	(JP) .	
8-286925		11/1996	(JP) .	
8-287195		11/1996	(JP) .	
11284684A	*	10/1999	(JP)	H04L/29/06

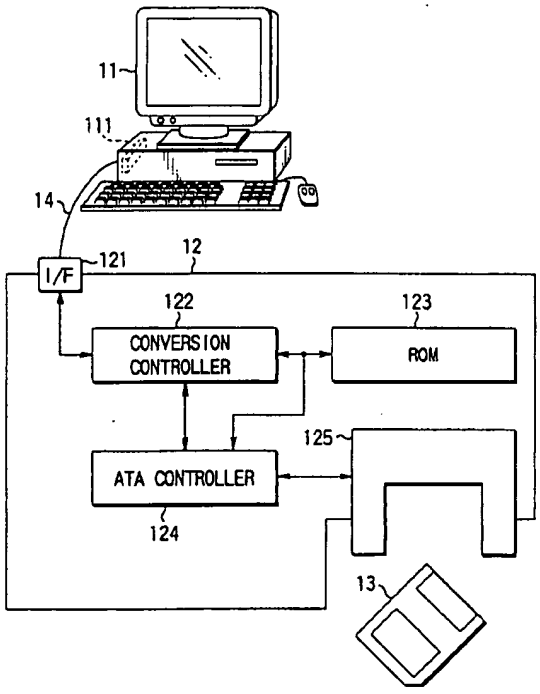
* cited by examiner

Primary Examiner—Thomas C. Lee
Assistant Examiner—Rehana Perveen
(74) *Attorney, Agent, or Firm*—Oblon, Spivak, McClelland, Maier & Neustadt, P.C.

(57) **ABSTRACT**

In order to access a memory card of the ATA specification, a computer generates a command based on the USB. A conversion controller in a reader/writer receives the command and converts it into a command of the ATA specification and supplies it to a controller of the ATA specification. The controller accesses the memory card based on the command of the ATA specification thus supplied. The conversion controller converts the formats of the data of the USB specification and the data of the ATA specification to each other. Consequently, the computer can access the memory card of the ATA specification constituting a conventional standard product using the USB of serial communication smaller in the area occupied by the connector.

20 Claims, 10 Drawing Sheets



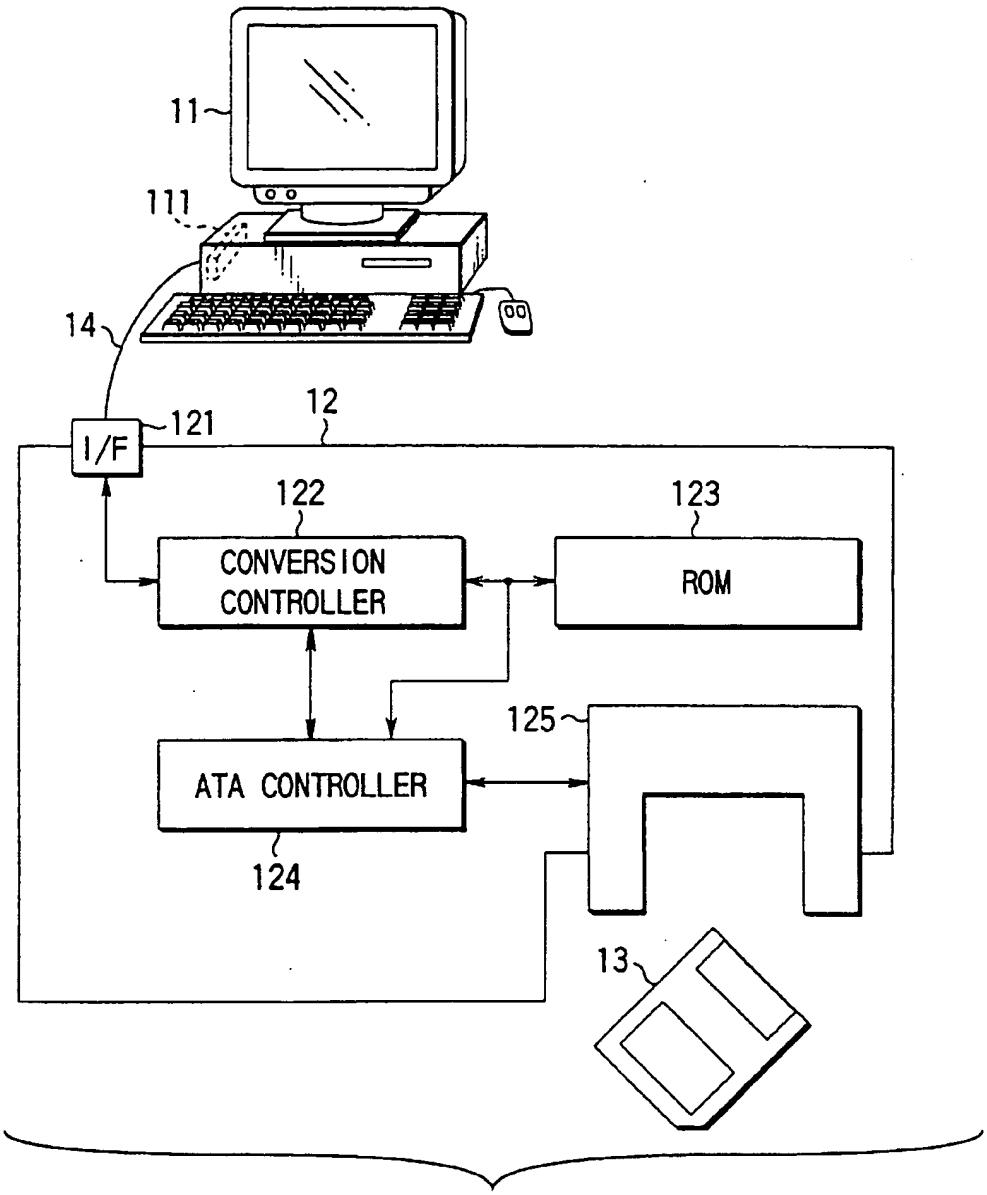

11
111
14
12
I/F
121
122
123
CONVERSION CONTROLLER
ROM
125
ATA CONTROLLER
124
13

FIG. 1

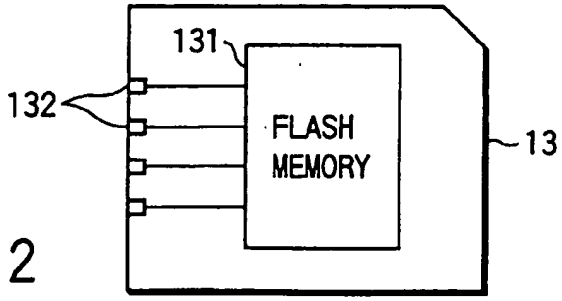

131
132
FLASH MEMORY
13

FIG. 2

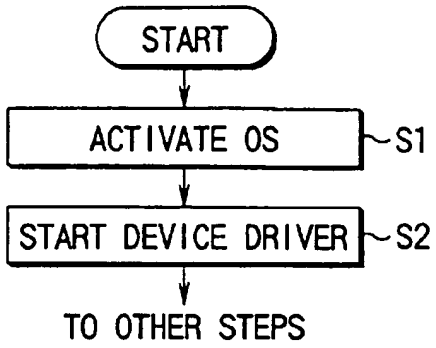

START
ACTIVATE OS
S1
START DEVICE DRIVER
S2
TO OTHER STEPS

FIG. 3

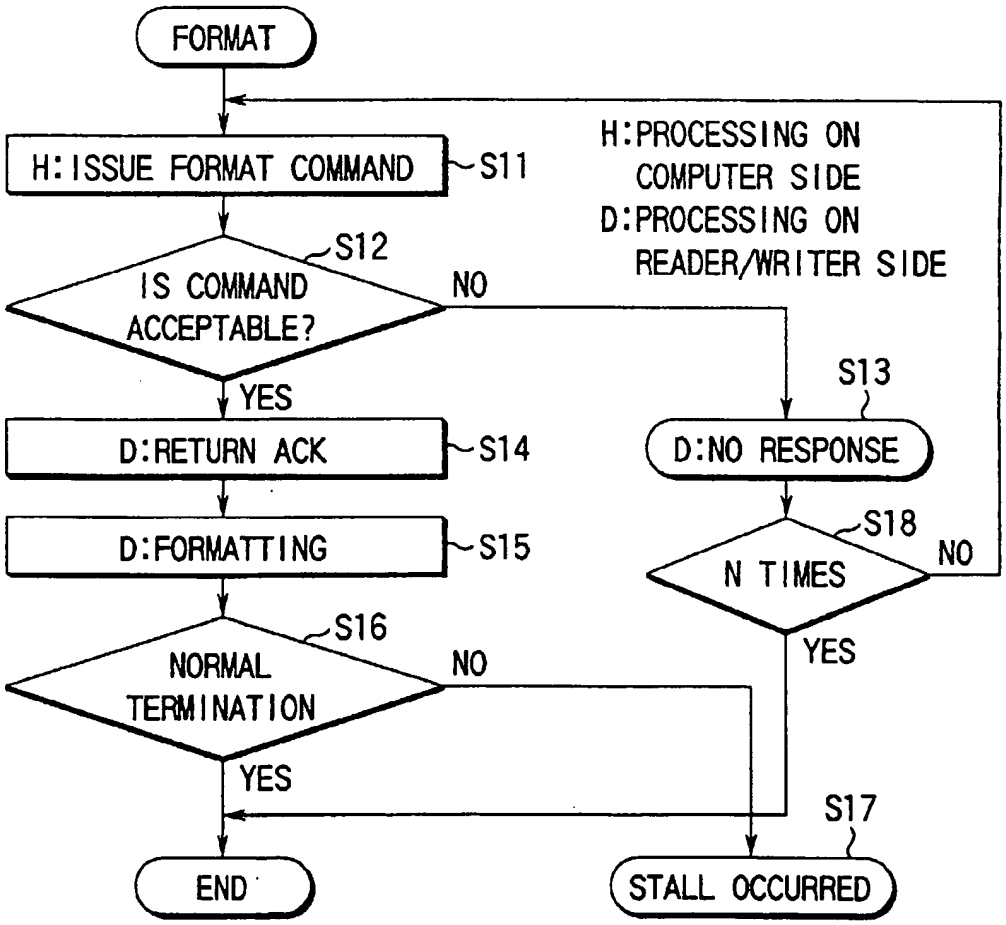

FORMAT
H:ISSUE FORMAT COMMAND
S11
H:PROCESSING ON COMPUTER SIDE
D:PROCESSING ON READER/WRITER SIDE
S12
IS COMMAND ACCEPTABLE?
NO
YES
S13
D:RETURN ACK
S14
D:NO RESPONSE
D:FORMATTING
S15
S18
N TIMES
NO
YES
S16
NORMAL TERMINATION
NO
YES
S17
END
STALL OCCURRED

FIG. 4

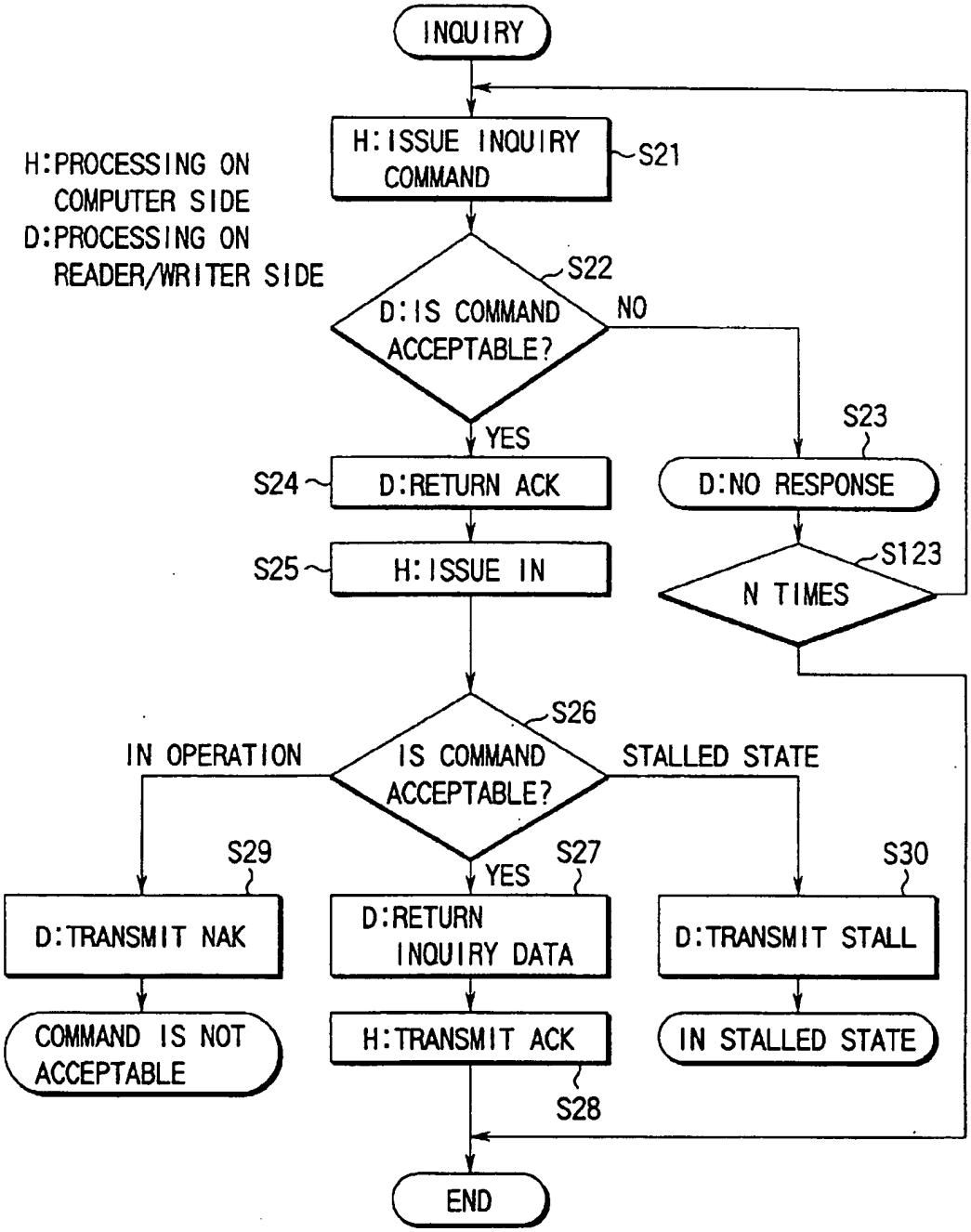

INQUIRY
H:ISSUE INQUIRY COMMAND
S21
H:PROCESSING ON COMPUTER SIDE
D:PROCESSING ON READER/WRITER SIDE
S22
D:IS COMMAND ACCEPTABLE?
NO
YES
S23
S24
D:RETURN ACK
D:NO RESPONSE
S25
H:ISSUE IN
S123
N TIMES
S26
IN OPERATION
IS COMMAND ACCEPTABLE?
STALLED STATE
S29
YES
S27
S30
D:TRANSMIT NAK
D:RETURN INQUIRY DATA
D:TRANSMIT STALL
COMMAND IS NOT ACCEPTABLE
H:TRANSMIT ACK
IN STALLED STATE
S28
END

FIG. 5

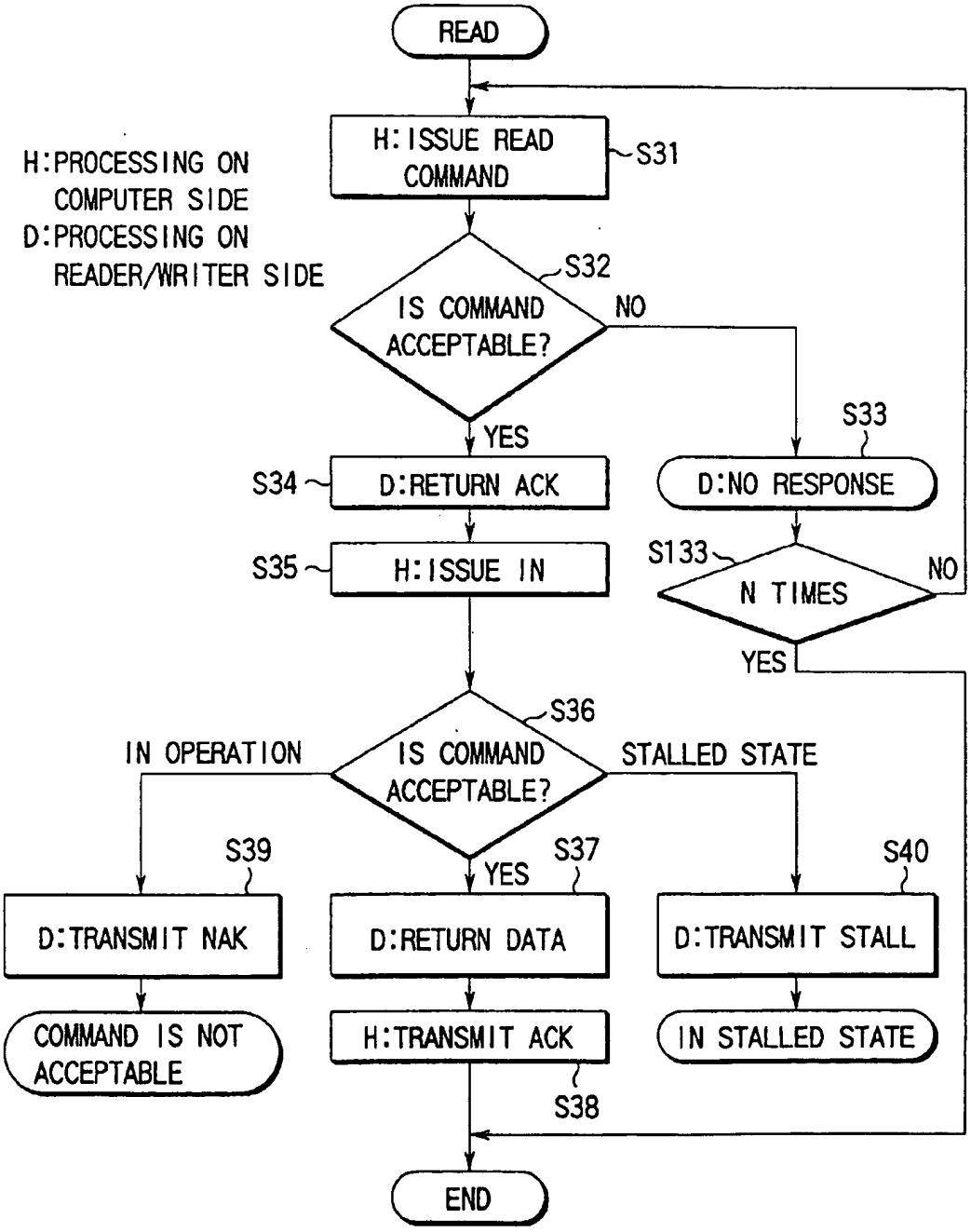

READ
H:PROCESSING ON COMPUTER SIDE
D:PROCESSING ON READER/WRITER SIDE
H:ISSUE READ COMMAND
S31
S32
IS COMMAND ACCEPTABLE?
NO
YES
S33
S34
D:RETURN ACK
D:NO RESPONSE
S35
H:ISSUE IN
S133
N TIMES
NO
YES
S36
IS COMMAND ACCEPTABLE?
IN OPERATION
STALLED STATE
S39
S37
S40
YES
D:TRANSMIT NAK
D:RETURN DATA
D:TRANSMIT STALL
COMMAND IS NOT ACCEPTABLE
H:TRANSMIT ACK
IN STALLED STATE
S38
END

FIG. 6

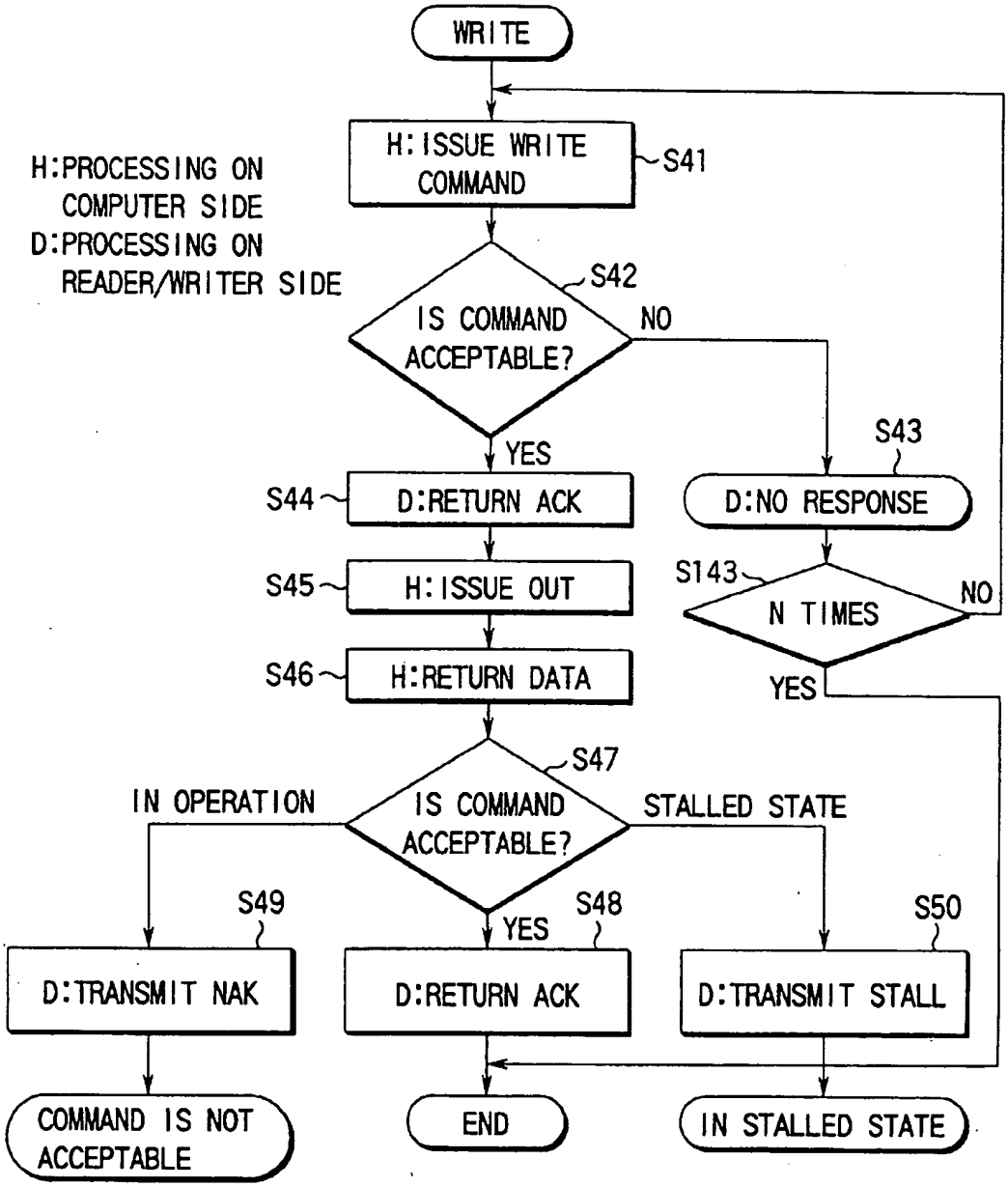

WRITE
H:PROCESSING ON COMPUTER SIDE
D:PROCESSING ON READER/WRITER SIDE
H:ISSUE WRITE COMMAND
S41
S42
IS COMMAND ACCEPTABLE?
NO
YES
S43
S44
D:RETURN ACK
D:NO RESPONSE
S45
H:ISSUE OUT
S143
N TIMES
NO
YES
S46
H:RETURN DATA
S47
IN OPERATION
IS COMMAND ACCEPTABLE?
STALLED STATE
S49
S48
S50
YES
D:TRANSMIT NAK
D:RETURN ACK
D:TRANSMIT STALL
COMMAND IS NOT ACCEPTABLE
END
IN STALLED STATE

FIG. 7

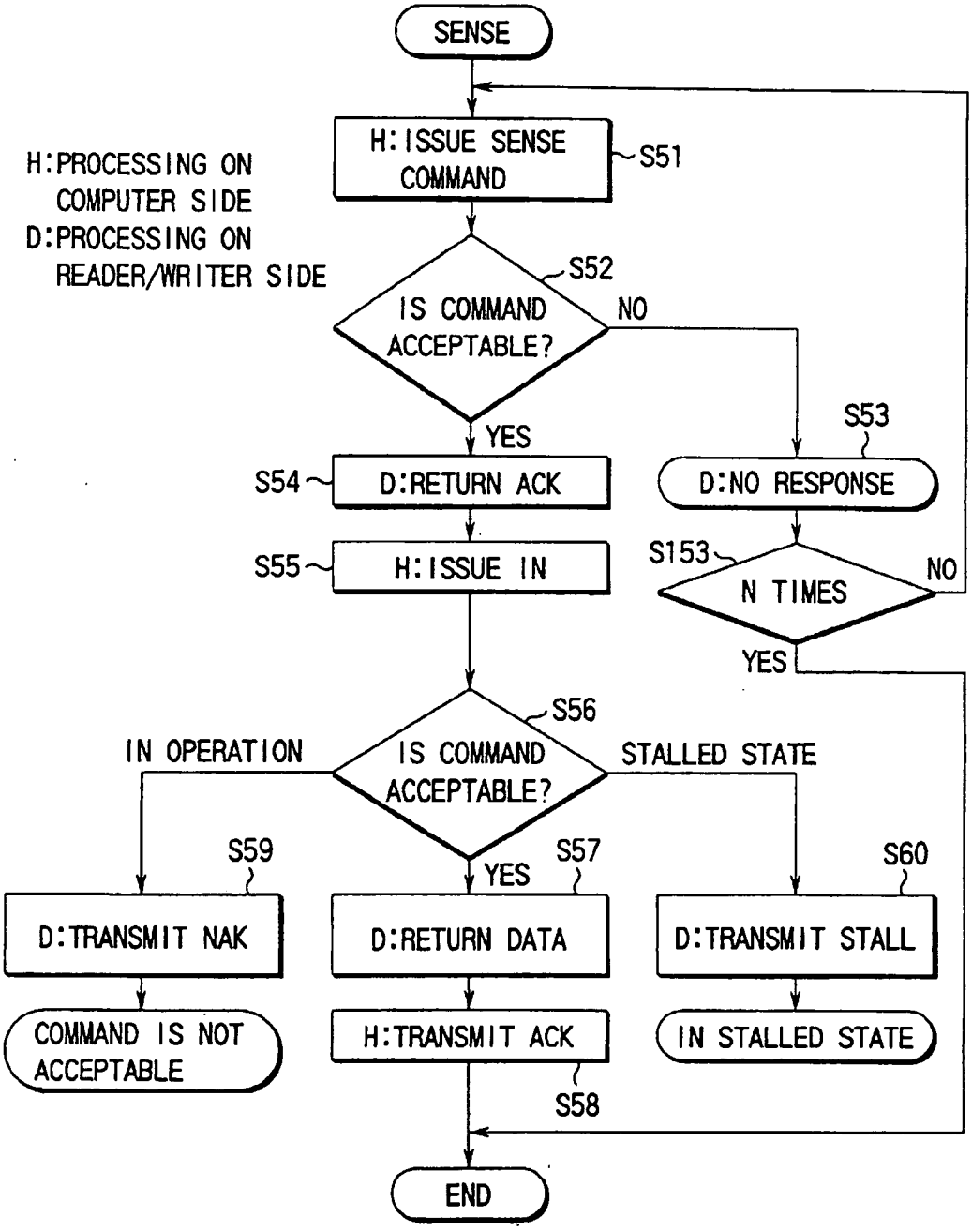

SENSE
H:PROCESSING ON COMPUTER SIDE
D:PROCESSING ON READER/WRITER SIDE
H:ISSUE SENSE COMMAND
S51
IS COMMAND ACCEPTABLE?
S52
NO
YES
S53
S54
D:RETURN ACK
D:NO RESPONSE
S55
H:ISSUE IN
S153
N TIMES
NO
YES
S56
IN OPERATION
IS COMMAND ACCEPTABLE?
STALLED STATE
S59
S57
S60
YES
D:TRANSMIT NAK
D:RETURN DATA
D:TRANSMIT STALL
COMMAND IS NOT ACCEPTABLE
H:TRANSMIT ACK
IN STALLED STATE
S58
END

FIG. 8

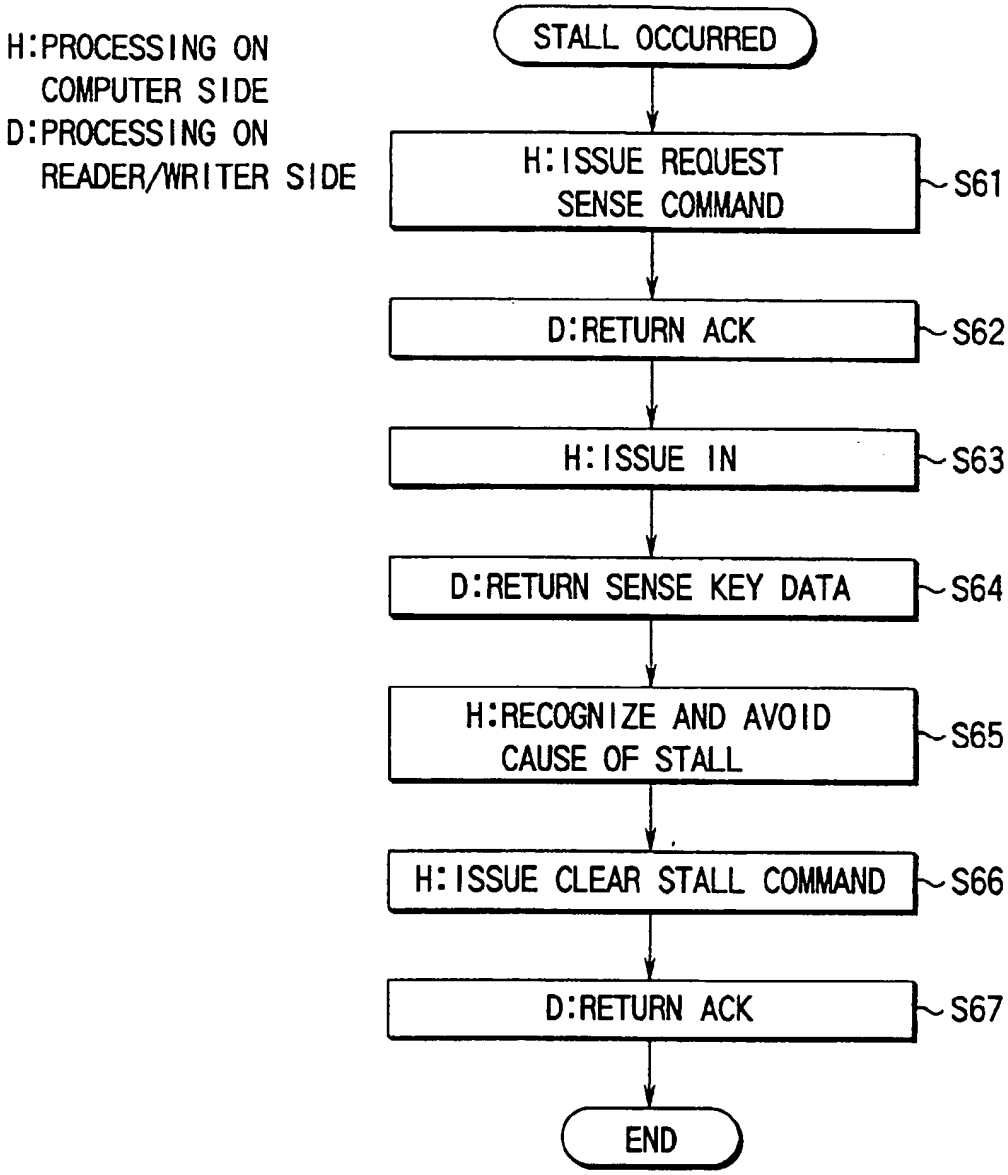

H:PROCESSING ON COMPUTER SIDE
D:PROCESSING ON READER/WRITER SIDE
STALL OCCURRED
H:ISSUE REQUEST SENSE COMMAND
S61
D:RETURN ACK
S62
H:ISSUE IN
S63
D:RETURN SENSE KEY DATA
S64
H:RECOGNIZE AND AVOID CAUSE OF STALL
S65
H:ISSUE CLEAR STALL COMMAND
S66
D:RETURN ACK
S67
END

FIG. 9

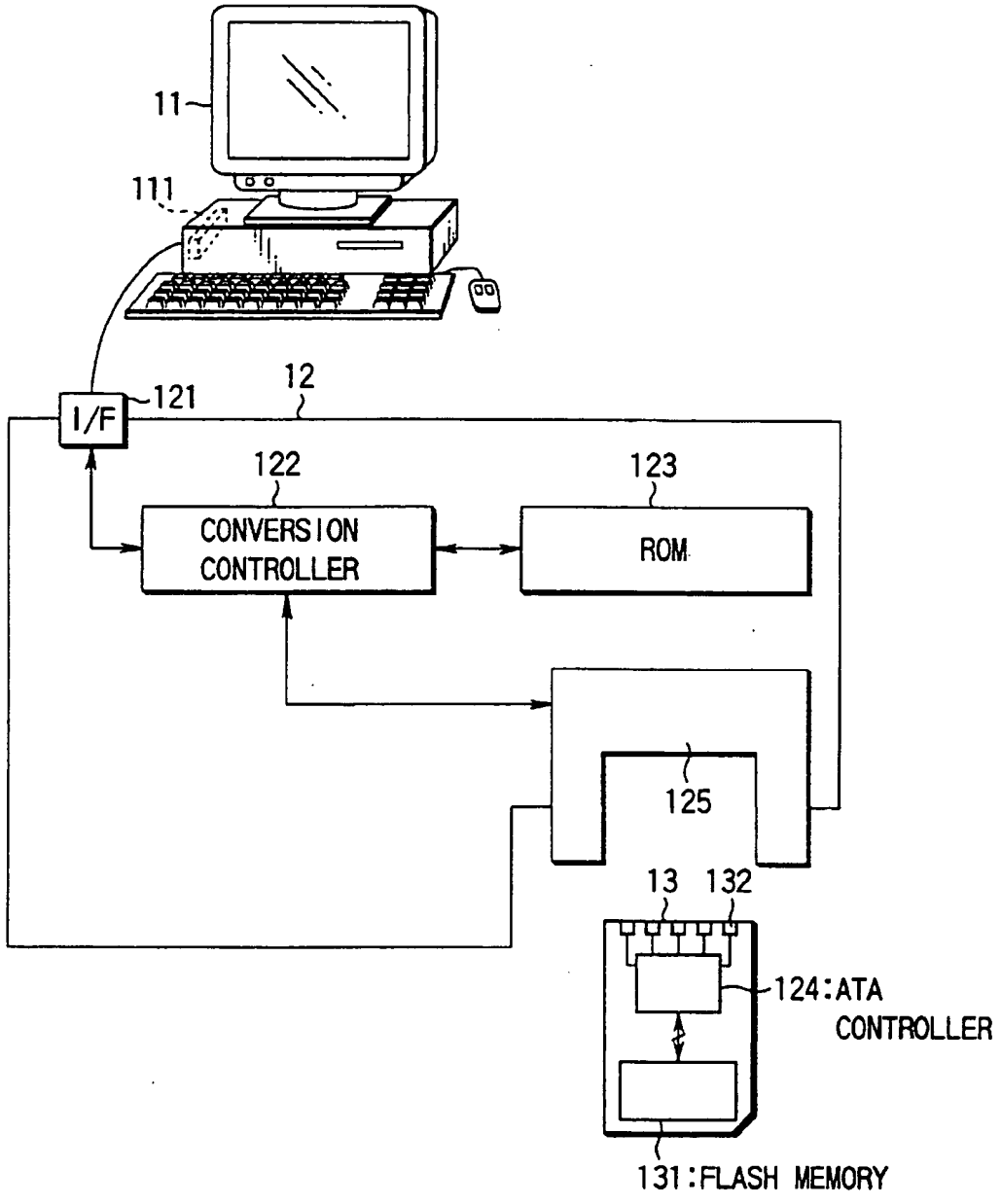

11
111
12
I/F
121
122
123
CONVERSION CONTROLLER
ROM
125
13
132
124:ATA CONTROLLER
131:FLASH MEMORY

FIG. 10

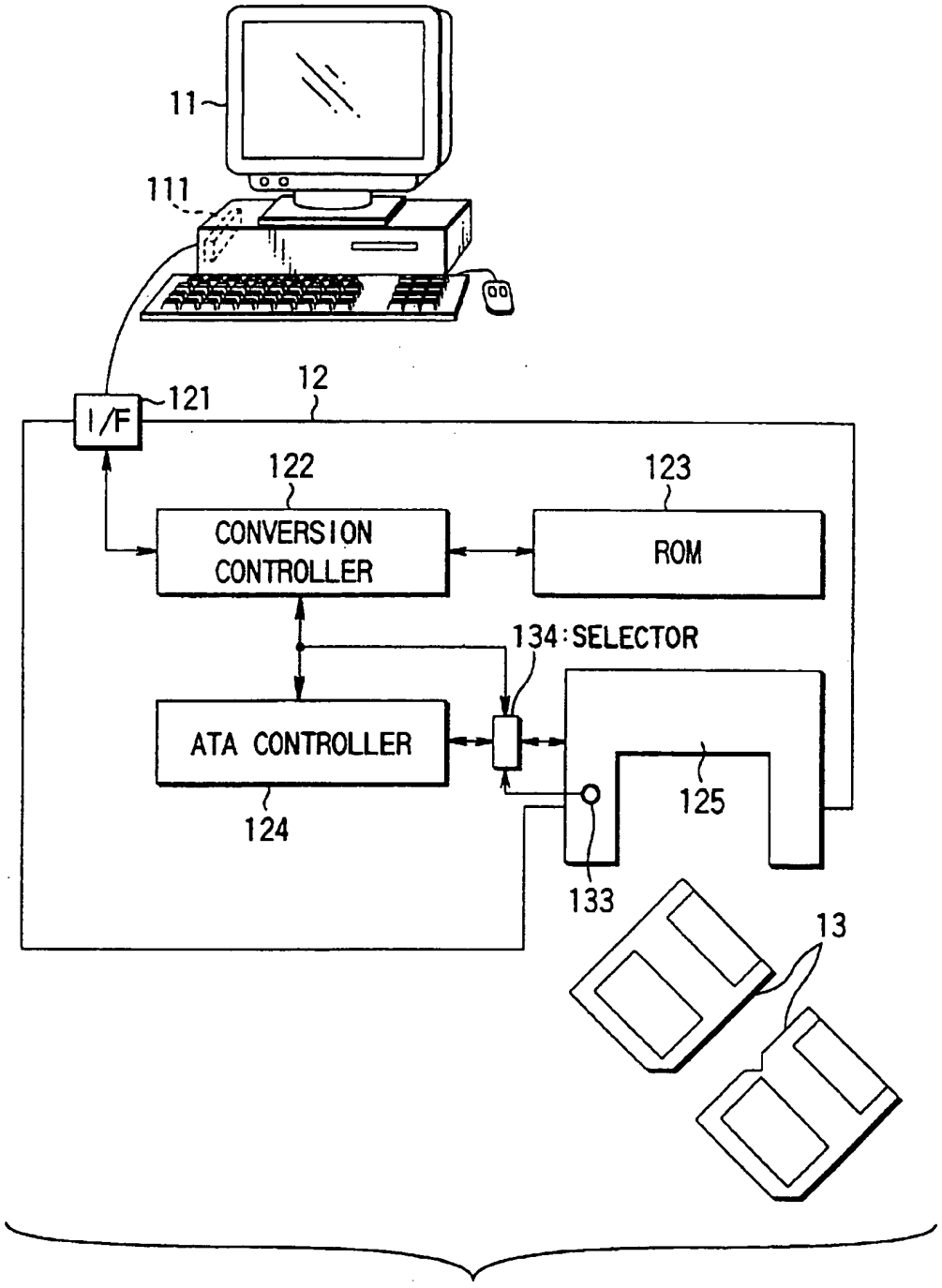

11
111
12
I/F
121
122
123
CONVERSION CONTROLLER
ROM
134:SELECTOR
ATA CONTROLLER
124
125
133
13

FIG. 11

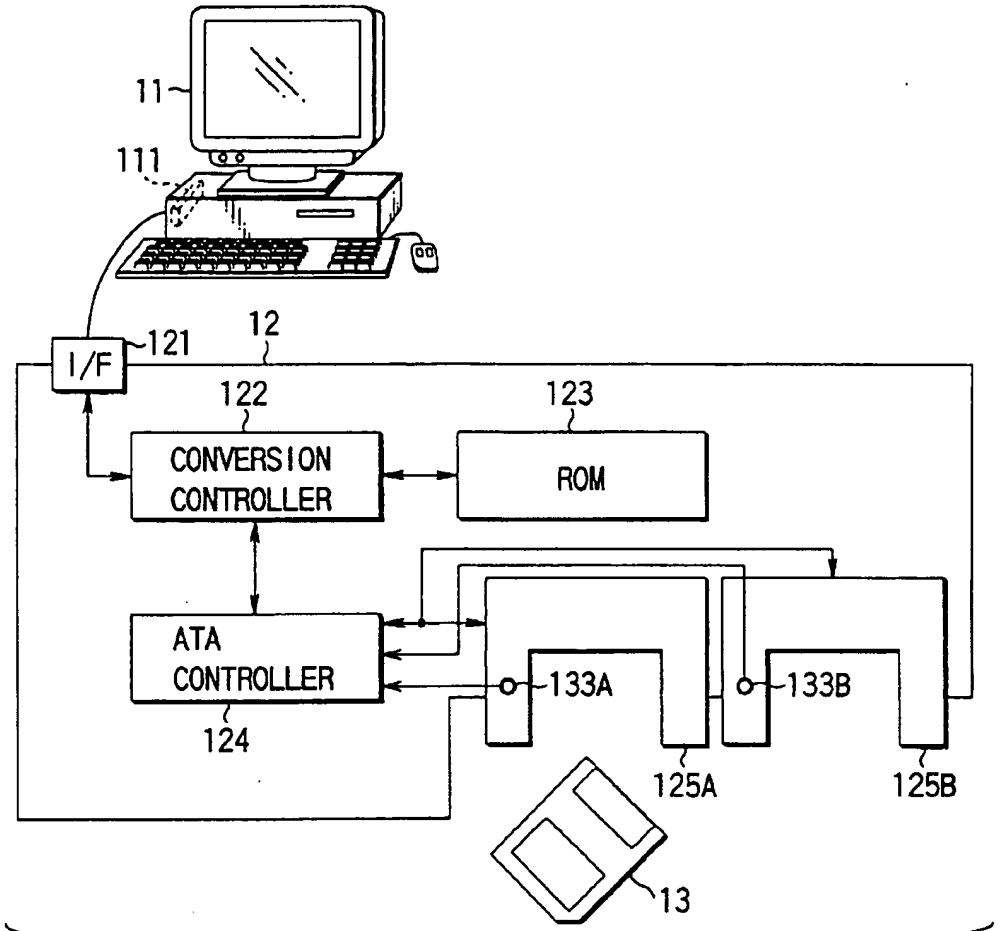

11
111
12
I/F
121
122
123
CONVERSION CONTROLLER
ROM
ATA CONTROLLER
124
133A
133B
125A
125B
13

FIG. 12

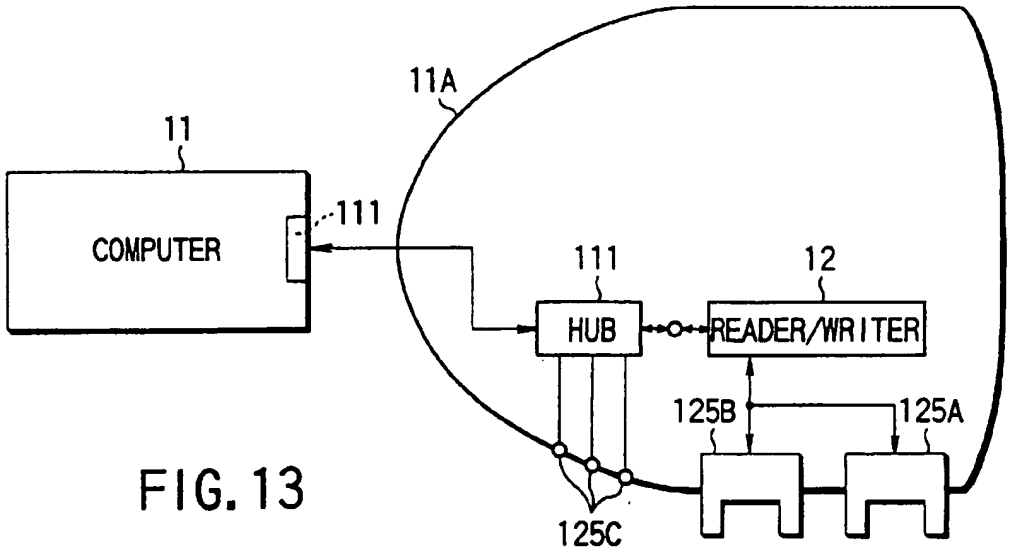

11A
11
COMPUTER
111
111
12
HUB
READER/WRITER
125B
125A
125C

FIG. 13

COMPUTER SYSTEM, EXTERNAL STORAGE, CONVERTER SYSTEM, AND RECORDING MEDIUM FOR CONVERTING A SERIAL COMMAND AND DATA STANDARD TO A PARALLEL ONE

BACKGROUND OF THE INVENTION

The present invention relates to a serial interface technique for a computer, particularly a technique for making the conventional external memory device of the ATA standard accessible with a command of the USB standard.

A floppy disk drive, a hard disk drive, etc. based on the ATA (AT Attachment) standard are known as conventional external memory devices. These memory devices pose the problem that they easily succumb to an external magnetism and the recorded data are liable to be lost. Also, these memory devices encounter the problem that they cannot be used for a portable terminal such as the PDA (Personal Data Assistance) operated by battery due to a large power consumption.

In order to solve these problems, a technique finds practical applications for recording and reproducing data by handling a memory card using a non-volatile memory such as a flash memory in the same way as the magnetic disk drive.

The conventional external memory device, which operates through a SCSI (Small Computer Serial Interface) constituting a parallel interface, has many pins for a connector. The connector is thus increased in size thereby making it difficult to secure the arrangement location of the connectors of the terminal and the external memory device.

It is considered to solve this problem by sending a command in a serial form from a computer, and processing the command by converting it into a parallel form within the external device. For example, a command based on SCSI is transmitted in a serial form and converted into a parallel form in the external memory device. However, in this case, the computer must process the command in a method different from the existing ones. The solution, therefore, is not practical.

On the other hand, a serial interface of the USB (Universal Serial Bus) standard has recently been proposed. A computer having an input/output port conforming with this standard is also in practical use. By using this standard input/output, data can be stored in and read from an external memory device at high speed. Therefore, data transmission and receipt to and from these cards using USB is expected. Since the command system and the protocol are quite different between USB and SCSI, however, the problem encountered is that the external memory device conforming with the SCSI constituting the conventional hardware and software resources cannot be used.

Also, there is another problem that the supplier of the external memory device must provide two types of system including SCSI and USB.

The above problem is not limited to the external memory device, and a similar problem arises when the communication is to be established between a computer and peripheral equipment thereof.

BRIEF SUMMARY OF THE INVENTION

An object of the present invention is to provide a system and a method by which a high-speed communication is made possible between a computer and peripheral equipment while minimizing the area occupied by the terminals on the computer and using the existing resources effectively.

Another object of the invention is to provide a technique by which the peripheral equipment can be controlled by a command based on the USB standard while maintaining the compatibility with the existing resources.

Still another object of the invention is to provide multi-purpose peripheral equipment.

According to a first aspect of the invention, there is provided a computer system comprising a computer including a computer-side serial interface for issuing a command based on the serial communication standard through the serial interface, a storage medium, a storage-side serial interface connected to the computer-side serial interface, and a converter for converting a command based on the serial communication standard supplied serially through the computer-side serial interface and the memory-side serial interface into a corresponding parallel command based on the parallel communication standard, and an external storage including an access section for controlling the access to the storage medium based on the parallel command supplied from the converter.

With this configuration, the computer establishes the communication through the serial interface. As compared with the case using a parallel interface, therefore, the terminal area can be reduced and thus the area occupied by the connector on the system can be reduced.

Also, the external storage is based on the parallel communication standard, and by converting a command based on the serial communication standard into a command based on the parallel communication command, the existing normal external storage can be directly accessed thereby maintaining the compatibility with the existing system.

The computer is not limited to the desktop computer or notebook-sized computer, but generally includes all computers for processing data by accessing the external storage of a PDA (Personal Data Assistance), a palmtop computer, a digital still camera, a portable telephone or the like.

The computer issues a command based on the USB (Universal Serial Bus) standard, for example, the converter of the external storage converts a command based on the USB standard into a corresponding parallel command based on the ATA standard, for example, the access section controls the access to the storage medium based on the parallel command supplied from the converter, for example.

The USB is a protocol for carrying out a high-speed serial communication. A high-speed communication becomes possible by complying with this standard. A simple compliance with this communication standard, however, makes it impossible to access the external storage based on the existing ATA standard. In view of this, a command issued by the computer is converted into a command based on the ATA standard and the storage medium is accessed. Consequently, the existing external storage based on the ATA standard can be used.

According to a second aspect of the invention, there is provided a memory device for a computer, comprising a converter configured to be connectable to a serial communication terminal of the computer for converting a command based on a first standard supplied serially from the computer into a parallel command based on a second standard different from the first standard, and an access section for controlling the access to a storage medium according to the parallel command based on the second standard supplied from the converter.

With this configuration, communication is made possible using a serial communication terminal of the computer. As a result, it becomes possible to minimize the area occupied

by the connector of the computer. Also, since the command based on the first standard supplied serially is converted into a command based on the second parallel standard, the compatibility is secured with the memory device for controlling the access to the storage medium based on the widely-used parallel command. The computer includes all the ones which access a storage medium and process the data thereof.

The first standard is the USB (Universal Serial Bus) standard, for example, and the second standard is the ATA (AT Attachment), for example. The USB standard is a protocol for high-speed serial communication. By compliance with this standard, high-speed communication becomes possible. Also, substantially all the existing external storages are based on the ATA standard. By converting the command issued by the computer into a command based on the ATA standard and accessing the storage medium, therefore, the compatibility with the existing memory devices can be maintained.

The converter can include one for converting the data format supplied serially based on the first standard from the computer into the parallel data of the format based on the second standard. The access section, on the other hand, can include a write section for writing the data of the format based on the second standard into the storage medium in response to a write command based on the second standard supplied from the converter.

Also, the access section reads the data stored in the recording medium and supplies it to the converter in the format based on the second standard in response to a read command based on the second standard supplied from the converter, and the converter can include a source for converting the data supplied from the access section into the serial data of the format based on the first standard and supplying it to the computer.

This configuration can secure the compatibility of data as well as commands.

As to each command from the computer for which a response can be returned to the computer without using the access section, the converter returns a response to the computer without converting the command into a command based on the second standard. Each command from the computer which requires access to the storage medium by the access section, on the other hand, is converted to a command based on the second standard and supplied to the access section.

The commands supplied from the computer include those commands requiring the processing by the access section and those commands which can be uniquely processed by the converter. This configuration can shorten the response time as the converter directly processes the commands that can be processed thereby.

The system can comprise a mounting member for mounting the storage medium thereon removably so that the storage medium thus mounted may be accessed.

In the process, the access section may access the storage medium arranged fixedly in the memory device and mounted on the mounting member or may be integrally formed with the storage medium and mounted on the mounting member.

Also, the storage medium can include a flash memory and the external storage may function substantially similarly to the magnetic disk drive.

According to a third aspect of the invention, there is provided a converter system comprising a first node based on a serial communication standard, a second node based on a parallel communication standard, a converter for converting a command based on the serial communication standard supplied serially through the first node into a corresponding parallel command based on the parallel communication standard in the case where the particular command requires access to the system based on the parallel communication standard and applying the resulting command to the second node, and a transmitter for transmitting through the first node a response to the command based on the serial communication standard supplied serially through the first node without converting it into a command based on the parallel communication standard in the case where the particular command requires no access to the system.

With this configuration, communication becomes possible between a serial port and a parallel port of different command systems, for example, through the converter system. Also, depending on the type of a particular command, the response time can be shortened since the response to the command is transmitted without converting the command.

The converter can include an output section for converting the format of the data based on the serial communication standard supplied through the first node into the data of the format based on the parallel communication standard, and producing the resulting data to the second node, and an output section for converting the format of the data based on the parallel communication standard supplied through the second node into the data of the format based on the serial communication standard and applying the resulting data to the first node.

This configuration makes possible data communication.

The first node is to be connected to a serial communication terminal based on the serial communication standard of the computer, for example, and the second node is connected to the access section for accessing the storage medium based on the command supplied from the converter, for example.

The second node is connected fixedly to the access section, for example, and the access section is connected to another access section for accessing the storage medium removably mounted. This configuration is applicable to the case in which a controller or the like for accessing the storage medium is provided in the converter system, for example, and a removable recording medium not including the controller is accessed.

Also, the access section can be connected removably to the second node to access the storage medium in the state so connected. This configuration is applicable to the case in which the storage medium is integrated with a controller or the like for accessing the storage medium and connected with the second node for access.

The converter and the transmitter include, for example, a memory for storing a program for converting the command based on the USB standard into the command based on the ATA standard and a program for responding to the command based on the USB standard, and a processor for receiving the command supplied through the first node and executing the program corresponding to the received command thereby to convert or respond to the command.

According to a fourth aspect of the invention, there is provided a medium for recording a program for causing a processor to execute the step of converting a command based on the USB standard into a command based on the ATA standard, the step of responding to the command based on the USB standard, the step of converting the data format based on the USB standard into the data of the format based on the ATA standard and the step of converting the data

based on the ATA standard into the data of the format based on the USB standard.

Additional objects and advantages of the invention will be set forth in the description which follows, and in part will be obvious from the description, or may be learned by practice of the invention. The objects and advantages of the invention may be realized and obtained by means of the instrumentalities and combinations particularly pointed out hereinafter.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING

The accompanying drawings, which are incorporated in and constitute a part of the specification, illustrate presently preferred embodiments of the invention, and together with the general description given above and the detailed description of the preferred embodiments given below, serve to explain the principles of the invention.

FIG. **1** is a block diagram showing a basic configuration of a computer system according to a first embodiment of the present invention;

FIG. **2** is a diagram showing a configuration of a memory card;

FIG. **3** is a flowchart showing the operation for activating the computer system;

FIG. **4** is a flowchart for explaining the formatting process;

FIG. **5** is a flowchart for explaining the inquiry process;

FIG. **6** is a flowchart for explaining the read process;

FIG. **7** is a flowchart for explaining the write process;

FIG. **8** is a flowchart for explaining the sense process;

FIG. **9** is a flowchart for explaining the stall process;

FIG. **10** is a block diagram showing a basic configuration of a computer system according to a second embodiment of the invention;

FIG. **11** is a block diagram showing a basic configuration of a computer system according to a third embodiment of the invention;

FIG. **12** is a block diagram showing a basic configuration of a computer system according to a fourth embodiment of the invention; and

FIG. **13** is a block diagram showing an application of a computer system according to the invention.

DETAILED DESCRIPTION OF THE INVENTION

An external storage device according to an embodiment of the invention will be explained below.

First Embodiment

FIG. **1** shows a configuration of a computer system according to a first embodiment of the invention.

According to FIG. **1**, the computer system comprises a computer **11**, a reader/writer (external storage) **12** and a removable memory card **13**.

The computer **11** includes a personal computer or the like having a serial interface **111** based on the USB standard. The computer **11** inputs and outputs various commands, control signals and data by way of the USB terminal in order to write, read, erase or otherwise process data with an external storage as a type of disk drive according to an OS (operating system) and a predetermined driving operation.

The memory card **13** includes terminals **132** on the surface thereof as shown in FIG. **2**, and an internal flash memory **131** connected to the terminals **132**. The memory card **13** functions as what is called a silicon disk or a PC card according to the ATA standard, and stores data and reads, outputs and erases the stored data under an external control.

The reader/writer **12** includes, as shown in FIG. **1**, a serial interface (USB interface) **121** based on the USB standard, a conversion controller **122**, a ROM **123**, an ATA (AT Attachment) controller **124**, and a connector **125**.

The USB interface **121** is a node connected to the computer **11** for transmitting and receiving data based on the USB standard to and from the computer **11**.

The conversion controller **122** is configured of a one-chip microprocessor or the like and operates in accordance with the program stored in the ROM **123**. The commands and data based on the USB standard supplied from the computer **11** are converted into the commands and data based on the ATA standard and outputted to the ATA controller **124**. The control signals and the data based on the ATA standard supplied from the ATA controller **124**, on the other hand, are converted into the control signals and the data based on the USB standard and supplied to the computer **11** through the USB interface **121**.

The ROM **123** stores a program, a fixed data or the like for defining the operation of the conversion controller **122**, or, for example, a program for converting a command based on the USB standard into a command based on the ATA standard. The commands based on the USB standard include a command required to access the memory card **13** and a command not required to access the memory card **13**. As for the command required to access the memory card **13**, the ROM **123** stores a program module for converting each command into a corresponding ATA command, while as for the command not required to access the memory card **13**, the ROM **123** stores a program module for responding to the particular command.

Also, the ROM **123** stores the program used by the conversion controller **122** to convert the data format based on the USB standard and the data format based on the ATA standard to each other.

The ATA controller **124** is a read/write controller based on the ATA standard and reads/writes data from and into the memory card **13**.

The connector **125** is a node for connecting the ATA controller **124** and the memory card **13** to each other, and includes a slot in which the memory card **13** is fitted removably, and a connecting terminal connected to the ATA controller **124** and the terminal **132** of the memory card **13** mounted.

Now, the operation of the computer system configured as described above will be sequentially explained with reference to the flowcharts of FIGS. **3** to **9**.

(Activation process)

First, upon activation of the computer **11**, as shown in FIG. **3**, the OS is activated (step S1) first of all. Then, a dedicated device driver is started for accessing the reader/writer **12** under the control of the OS (step S2).

After that, the computer **11** starts other required programs appropriately and transfers to the initialized state.

(Format of memory card **13**)

For the computer **11** to use the memory card **13** as an external storage, the memory card **13** is required to be formatted in accordance with the OS standard.

This formatting operation is performed in compliance with an instruction from the operator, for example. First, the operator opens the window of the device driver and instructs the memory card **13** to be formatted by way of an input unit

like a keyboard or a mouse. In response to this instruction, the device driver issues a formatting command as shown in FIG. 4 (step S11). This formatting command is based on the USB standard.

This formatting command is transmitted to the conversion controller 122 through the USB terminal of the computer 11, the cable 14 and the USB interface 121.

The conversion controller 122, if unable to accept the formatting command for some reason, for example, because other process is under execution, ignores and fails to respond to the particular command (steps S12, S13). In this case, if the formatting command, after N transmissions, or after three trials, for example, fails to be accepted, then the process is terminated (step S18).

The conversion controller 122, on the other hand, if capable of accepting the formatting command, transmits an ACK (acknowledge) signal to the computer 1 (steps S12, S14).

The conversion controller 122 determines that the received command is a formatting command, reads the program module for defining the processing of the formatting command based on the USB standard out of the ROM 123, and executes the formatting process in accordance with the program (step S15).

In this formatting process, the conversion controller 122 issues a FLASH READ command to the ATA controller 124. The ATA controller 124 executes the FLASH READ command and detects a stalled block. Then, the ATA controller 124 issues a BLOCK ERASE command to erase all the blocks other than the stalled block. The ATA controller 124 thus issues a FLASH WRITE command and writes the initialization data such as CIS (Card Information Structure).

Upon normal completion of formatting, the ATA controller 124 transmits a normal termination signal to the conversion controller 122. In response to the normal termination signal, the conversion controller 122 outputs a normal termination signal based on the USB standard to the computer 11 (step S16). In response to the normal termination signal, the device driver on the computer 11 notifies the operator of the termination of the formatting or otherwise performs a predetermined process.

In the case where the formatting of the memory card 13 fails to be terminated normally for some reason or other, on the other hand, the ATA controller 124 transmits an abnormal termination signal to the conversion controller 122. In response to the abnormal termination signal, the conversion controller 122 outputs an abnormal termination signal based on the USB standard to the computer 11 while at the same time entering a stalled state (step S17). In this stalled state, the main bus is closed, and the conversion controller 122 responds only a stall to the commands other than a set-up packet. In the case of a stall, the device driver of the computer 11 issues a REQUEST SENSE command described later, grasps the contents of the error, issues a CLEAR STALL command (clear feature) and restores the idle state from the stalled state.

The memory card 13 having a flash memory built therein may be formatted only at the time of issue of the memory card 13 but not by the reader/writer 12. In such a case, the conversion controller 122, upon receipt of a FORMAT command from the computer 11, returns an ACK signal to the computer 11. After that, however, the conversion controller 122 performs no special formatting operation.

(Acquisition of device information)

In the case where the computer system accesses the memory card 13, it is necessary to identify the external storage (device type, ISO version, response data format, product ID, etc.). In such a case, the system executes the process shown in FIG. 5. First, the device driver issues an INQUIRY command (step S21). This command is a serial command based on the USB standard.

This INQUIRY command is outputted through the USB interface (serial interface) of the computer 11, and transmitted through the USB interface 121 to the conversion controller 122.

The conversion controller 122, if unable to accept the INQUIRY command for some reason, ignores the command and fails to respond (steps S22, S23). In such a case, unless the formatting command is accepted after N transmissions thereof, or after three trials, for example, then the process is terminated (step S123).

The conversion controller 122, if able to accept the INQUIRY command, on the other hand, transmits the ACK signal to the computer 11 (steps S22, S24). Also, the conversion controller 122 acquires such information stored in the ROM 123 in advance as the device type of the external storage, the ISO version, the ECMA version, the ANSI version, the response data format, the additional data length, the vendor ID, the product ID and the product version.

In response to the ACK signal, the device driver on the computer 11 issues an IN command for requesting the data acquisition (step S25). This IN command is transmitted through the USB I/F 121 to the conversion controller 122.

The conversion controller 122, when capable of executing the IN command, transmits the device information acquired at step S24 to the computer 11 (steps S26, S27). The device driver on the computer 11, upon receipt of the INQUIRY data, transmits the ACK signal to the conversion controller 122 (step S28), and thus terminates the processing of the inquiry.

From the data thus supplied, the device driver acquires such information as the device type of the memory card 13, the ISO version, the ECMA version, the ANSI version, the response data format, the additional data length, the vendor ID, the product ID and the product version. These information are used for accessing the memory card 13 subsequently.

The conversion controller 122, while unable to execute the IN command issued by the computer 11 as other process is under execution, transmits a NAK signal to the computer 11 and notifies that the particular command cannot be executed (steps S26, S29).

In the case where the normal communication is impossible between the computer 11 and the conversion controller 122 for some reason, the stall signal based on the USB standard is issued while at the same time setting the system in the stalled state (steps S26, S30).

(Read process)

Now, the process of reading the data stored in the memory card 13 will be explained with reference to FIG. 6. In this case, the device driver issues a READ command in response to a request from an application program or the OS (step S31). This command includes the address, the byte length (number of bytes) and the amount of the data and is based on the USB standard.

This READ command is transmitted through the USB I/F 121 to the conversion controller 122.

The conversion controller 122, if unable to accept this READ command for some reason, ignores the command and fails to respond to it (steps S32, S33). In such a case, unless the formatting command is accepted after N transmissions, or after three trials, for example, then the process is terminated (step S133).

The conversion controller 122, if able to accept the READ command, on the other hand, transmits the ACK signal to the computer 11 (steps S32, S34).

Also, the conversion controller **122** reads from the ROM **123** a program module defining the process of converting the READ command based on the USB standard to the READ command based on the ATA standard, and in accordance with this program, converts the READ command based on the USB standard into the READ command based on the ATA standard and supplies the resulting command to the ATA controller **124** in parallel.

In response to the READ command thus converted, the ATA controller **124** reads the data stored in the corresponding address of the memory card **13** by a specified length and supplies it to the conversion controller **122**.

In response to the ACK signal from the conversion controller **22**, the device driver issues an IN command for requesting data acquisition (step **S35**). This IN command is transmitted through the USB I/F **21** to the conversion controller **122**.

The conversion controller **122**, when capable of executing the IN command, reads from the ROM **123** a program module defining the process for converting the data based on the ATA standard to the data based on the USB standard. In accordance with this program, the conversion controller **122** converts the format of the data based on the ATA standard supplied from the ATA controller **124** into the format based on the USB standard and supplies it to the computer **11** (steps **S36, S37**). The device driver, upon receipt of the data, transmits the ACK signal to the conversion controller **122** (step **S38**) and thus terminates the read process.

The data acquired in this way are supplied to an application or the OS and used for appropriate processing.

In the case where the IN command issued by the computer **11** cannot be executed as another process is under execution, the conversion controller **122** transmits the NAK signal to the computer **11** and notifies that the particular command cannot be executed (steps **S36, S39**).

In the case where normal communication is impossible between the computer **11** and the conversion controller **122** for some reason or other, the STALL signal based on the USB standard is outputted while at the same time setting the system in stalled state (steps **S39, S40**).

(Write process)

Now, the process for writing data in the memory card **13** will be explained with reference to FIG. **7**.

In this case, in response to a request from an application program or the OS, the device driver issues a WRITE command (step **S41**). This command includes the data address, the byte length (number of bytes), etc. and is based on the USB standard.

This WRITE command is transmitted through the USB I/F and the USB IF **121** of the computer **11** to the conversion controller **122**.

The conversion controller **122**, if unable to accept the WRITE command, ignores the command and fails to respond to it (steps **S42, S43**). In this case, unless the formatting command is accepted after N transmissions, or after three trials, for example, the process is terminated (step **S143**).

The conversion controller **122**, if ready to accept the WRITE command, transmits the ACK signal to the computer **11** (steps **S42, S44**).

Further, the conversion controller **122** reads from the ROM **123** a program module defining the process of converting the WRITE command based on the USB standard into the WRITE command based on ATA standard. In accordance with this program, the conversion controller **122** converts the WRITE command based on the USB standard supplied from the computer **11** into the WRITE command based on the ATA standard and transmits the resulting command to the ATA controller **124**.

The ATA controller **124** waits for the receipt of data in response to the WRITE command based on the ATA standard.

On the other hand, the device driver issues an OUT command instructing a data output in response to the ACK signal from the conversion controller **122** (step **S45**).

Furthermore, the data to be written are transmitted to the conversion controller **122** (step **S46**).

The conversion controller **122**, whenever capable of executing the OUT command (step **S47**), reads from the ROM **123** a program module defining the process for converting the format of the data based on the USB standard into the format based on the ATA standard. In accordance with this program, the conversion controller **122** converts the format of the data based on the USB standard supplied from the computer **11** into the format based on the ATA standard, and transmits the resulting data to the ATA controller **124**.

The ATA controller **124**, upon receipt of the data, writes the supplied data sequentially in the memory card **13**. The ATA controller **124**, upon completion of the data write operation, transmits a signal indicating the completion of the write operation to the conversion controller **122**, which in turn transmits the ACK signal to the computer **11** (step **S48**), thereby terminating the write process.

On the other hand, the conversion controller **122**, if unable to execute the OUT command issued by the computer **11** as another process is under execution, transmits the NAK signal to the computer **11** and notifies that the particular command cannot be executed (steps **S47, S49**).

In the case where normal communication fails between the computer **11** and the conversion controller **122** for some reason, the STALL signal based on the USB standard is outputted while at the same time setting the system in stalled state (steps **S47, S50**).

(State determination (sense) process)

Now, the sense process in which the device driver determines the state of the reader/writer **12** including the memory card **13** will be explained with reference to FIG. **8**.

In this case, the device driver issues a SENSE command (step **S51**). The SENSE command is of two types, a MODE SENSE command for making an inquiry about the storage capacity of the memory card **13** and a REQUEST SENSE command for making an inquiry about the state of the system. The device driver issues one of them, as required. This command is based on the USB standard.

The command thus issued is transmitted to the conversion controller **122** through the USB interface and the USB interface **121** of the computer **11**.

The conversion controller **122**, if unable to accept the SENSE command, ignores it and fails to respond to it (steps **S52, S53**). In this case, unless the formatting command is accepted after N transmissions, or after three trials, for example, then the process is terminated (step **S153**).

The conversion controller **122**, if ready to accept the SENSE command, on the other hand, transmits the ACK signal to the computer **11** (steps **S52, S54**).

Also, in the case where the SENSE command is the MODE SENSE command, the conversion controller **122** reads from the ROM **123** a program module defining the process for converting the MODE SENSE command based on the USB standard into the MODE SENSE command based on the ATA standard. In accordance with this program, the conversion controller **122** converts the MODE SENSE command based on the USB standard into an IDENTIFY

DRIVE command based on the ATA standard, and supplies the resulting command to the ATA controller **124** in parallel.

The ATA controller **124** reads a list of parameters such as capacity stored at a predetermined position in the memory card **13** in response to the IDENTIFY DRIVER command thus converted, and supplies the data to the conversion controller **122**.

The device driver, on the other hand, issues an IN command for requesting for acquisition of data in response to the ACK signal from the conversion controller **122** (step S55).

The conversion controller **122**, when capable of executing the IN command, reads from the ROM **123** a program module defining the process for converting the data based on the ATA standard into the data based on the USB standard. In accordance with this program, the conversion controller **122** converts the format of the parameter list based on the ATA standard supplied from the ATA controller **124** into the format based on the USB standard and transmits the result to the computer **11** (steps S56, S57). The device driver, upon receipt of the data, transmits the ACK signal to the conversion controller **122** (step S58), and terminates the sense process. The data acquired in this way is supplied to an application or the OS and used for accessing the memory card **13**.

In the case where the SENSE command is a REQUEST SENSE command, the conversion controller **122** reads from the ROM **123** a program module defining the process for converting the REQUEST SENSE command based on the USB standard into the IDENTIFY DRIVER command based on the ATA standard. In accordance with this program, the conversion controller **122** converts the REQUEST SENSE command based on the USB standard into the IDENTIFY DRIVER command based on the ATA standard, and supplies the resulting command in parallel to the ATA controller **124**.

The conversion controller **122** and the ATA controller **124** check the various parts in the system and collect the sense key data indicating the state of each part. The contents of the data thus sensed include whether or not an error exists in the memory, whether or not the memory is accessible, whether or not there exists an irreparable error, whether or not there exists a hardware error, and whether or not the data are protected. The ATA controller **124** supplies the collected sense key data to the conversion controller **122**.

In the case where the IN command issued by the computer **11** cannot be executed, on the other hand, the conversion controller **122** transmits the NAK signal to the computer **11** notifying that the particular command cannot be executed (steps S56, S59).

In the case where normal communication is impossible between the computer **11** and the conversion controller **122** for some reason, on the other had, the STALL signal based on the USB standard is outputted and the system is set in stalled state at the same time (steps S56, S60).

(Insertion and removal of memory card **13**)

In order to make possible the plug and play function by the OS, the ATA controller **24** checks the insertion and removal of the connector **125** into and from the memory card **13** at regular time intervals. Each time of insertion and removal, the fact is identified. In the case where there is any inquiry from the computer **11** through the conversion controller **122**, the result of identification is notified through the conversion controller **122** in interrupt transfer mode. The OS starts the plug and play process in response to this notification.

(Stall process)

Now, an explanation will be given of the process executed at the time when the communication path (cable) is stalled with reference to FIG. **9**. In such a case, the driver issues a REQUEST SENSE command (step S61).

In response to the REQUEST SENSE command, the conversion controller **122** transmits the ACK signal to the computer **11** (step S62).

The conversion controller **122** and the ATA controller **124** check each part in the system and collect the sense key data indicating the state of each part. The computer **11**, on the other hand, outputs an IN command in response to the ACK signal (step S63).

The conversion controller **122** transmits the sense key data collected at step S62 to the computer **11** in response to the IN command (step S64).

The device driver supplies the supplied sense key data to the OS and recognizes and avoids the cause of the stall (step S65). Further, the OS issues a CLEAR STALL command through the device driver (step S66). The conversion controller **122**, upon receipt of the command, restores the idle state from the stalled state and transmits the ACK signal to the computer **11** (step S67). Then, normal communication is made possible between the computer **11** and the external memory device **12**.

As described above, according to this embodiment, the external memory device can be accessed using a USB port constituting a serial input/output terminal of the computer **11**. As a result, the connector size poses no problem even in the case where the computer is such a small-sized equipment as the palmtop computer, the PDA (personal data assistance), the digital still camera or the portable telephone.

Also, the USB command used by the computer and the ATA command used by the conventional external memory device are interchangeably converted to each other. Thus, the conventional storage medium based on the ATA standard can be used directly and the compatibility with the existing systems can be maintained. Further, the use of the USB interface permits a high-speed communication.

Second Embodiment

In the first and second embodiments, the ATA controller **124** for controlling the memory card **13** is arranged in the reader/writer **12**. Alternatively, the ATA controller can be arranged in the memory card **13**. A configuration of such an example is shown in FIG. **10**. The operation of this example is identical to the operation of the first embodiment except that the communication between the conversion controller **22** and the ATA controller **124** (the communication based on the ATA standard) is established through the connector **125**.

It is also possible to arrange the SCSI/F **127** and the selector **128** according to the second embodiment in such a manner that the output terminal of the selector **128** is connected to the ATA controller **124** in the memory card **13** through the connector **125**.

Third Embodiment

The reader/writer **12** according to the first embodiment is used exclusively for the memory card **13** having no ATA controller built therein, and the reader/writer **12** according to the second embodiment is used exclusively for the memory card **13** with the ATA controller built therein. In place of them, a reader/writer that can be shared by either type of memory card can be provided.

An example configuration of this type of reader/writer **12** is shown in FIG. **11**. In this configuration, a sensor **133**

determines the type of the memory card **13** which is mounted on the connector **125**. In the case where the result of determination indicates that the memory card **13** includes no ATA controller, a selector **134** connects the ATA controller **124** and the connector **125**. In the case where the memory card **13** includes the ATA controller, on the other hand, the selector **134** connects the conversion controller **122** and the connector **125**.

With this configuration, the process can be executed in the reader/writer **12** regardless of the type of the memory card. The memory card **13** has formed therein the information indicating the type thereof by an opening or a notch. The sensor **133** includes an optical sensor or a microswitch.

Fourth Embodiment

In the first to third embodiments, the reader/writer **12** including only one memory card **13** and removed was explained. Instead, an arrangement is possible in which a plurality of memory cards **13** can be selectively mounted and removed.

As shown in FIG. **12**, the connector **125** includes a plurality of slots **125A**, **125B**. Each slot has sensors **133A**, **133B** arranged therein. The ATA controller **124** is supplied with a signal indicating a particular slot in which the memory card **13** is mounted. The conversion controller **122** makes an inquiry appropriately at the ATA controller **124** about the configuration of the system. The conversion controller **122**, upon receipt of an INQUIRY command from the computer **11**, notifies the computer **11** of the system configuration fetched in advance.

The computer **11** specifies and accesses the device (memory card **13**), for example, based on the system configuration thus notified.

Also, a plurality of memory cards **13** containing the ATA controller **124** can be mounted.

Further, the memory card **13** having the ATA controller **124** built therein and the memory card **13** having no ATA controller **124** built therein can be configured as a common memory card. In such a case the sensors **133A**, **133B** determine which type of memory card is mounted in which slot and notifies the result to the computer **11**. The computer **11** specifies the memory card **13** to be accessed, and issues a read/write command or the like. In the case where the designated memory card **13** has the controller **124** built therein, the conversion controller **122** supplies the ATA command directly to the particular memory card. The memory card **1** having no controller **124** built therein, if accessed, is done so through the ATA controller **124** arranged in the reader/writer **12**.

The reader/writer **12** is not required to be connected to an exclusive USB terminal. As shown in FIG. **13**, for example, a hub **111** connected to the USB terminals of the computer **11** is arranged in a housing of a display unit **11A** such as a CRT, and a plurality of the USB terminals can be led out of the hub **111** with one of them connected to the reader/writer **12**.

In the process, the slots **125A**, **125B** of one or a plurality of connectors **125** of the reader/writer **12** can be arranged at the front lower end or on the side of the display unit. The USB connection terminal **125C** also can be arranged in similar fashion.

With this configuration, the memory can be very conveniently used by inserting and removing the memory card **13** into and from the slot formed in the CRT or the like. Especially, the handling is facilitated by supporting the plug and play function. Also, in the case where the storage medium is a flash memory, the magnetic field which may be generated by the CRT has no adverse effect and the contents of memory can be held in stable fashion.

The present invention is not confined to the above-mentioned embodiments, but various modifications and applications are possible. In the embodiments, for example, the reader/writer **12** is controlled by the device driver operating on the OS. However, the reader/writer **12** can alternatively be controlled by the OS itself.

Also, the above-mentioned embodiments refer to the case in which the commands and data based on the USB standard and the commands and data based on the ATA standard are converted from each other. In spite of this, the invention is applicable with equal effect to the case where different serial and parallel standards are converted to each other. Also, the present invention is not limited to the case in which the external memory device of the computer is controlled, but is widely applicable to the case in which the peripheral equipment of the computer are controlled.

The whole or part of the programs stored in the device driver and the ROM **123** may be distributed by being stored in a recording medium (floppy disk, CD-ROM, etc.), so that the programs may be installed on the computer **11** or stored in the ROM **123**.

As described above, according to this invention, the dedicated area of the terminal of the computer or the like is minimized, and while maintaining the compatibility with the existing recording medium, data can be held and read out at high speed.

As described above, a computer system according to the present invention is not limited to the desktop computer or the note-sized personal computer but is applicable to computers in general for processing data by accessing an external memory device such as the PDA (personal data assistance), the palmtop computer, the digital still camera and the portable telephone.

Additional advantages and modifications will readily occur to those skilled in the art. Therefore, the invention in its broader aspects is not limited to the specific details and representative embodiments shown and described herein. Accordingly, various modifications may be made without departing from the spirit or scope of the general inventive concept as defined by the appended claims and their equivalents.

What is claimed is:

1. A computer system comprising:

a computer including a computer-side serial interface for issuing at least one command and data based on a serial communication standard through said computer-side serial interface;

a storage-side serial interface connected to said computer-side serial interface;

conversion means having a ROM storing a program m for receiving the at least one command and data based on said serial communication standard supplied through said computer-side serial interface and said storage-side serial interface and converting the at least one command and data into a corresponding at least one parallel command and data based on a parallel communication standard, said conversion means automatically converting the received at least one command and data based on said serial communication standard into the at least one parallel command and data based on the parallel communication standard in accordance with the program stored in said ROM; and

an external storage having access means and including a removable storage medium, said access means controlling access to said removable storage medium in accordance with the at least one parallel command and data based on the parallel communication standard supplied from said conversion means.

2. A computer system according to claim 1, wherein said computer issues the at least one command based on a USB (Universal Serial Bus) serial communication standard, said conversion means converts a command based on said USB standard into at least one corresponding parallel command based on an ATA (AT Attachment) parallel communication standard.

3. A computer system according to claim 1, wherein said ROM stored program is a program for converting at least one command based on a USB (Universal Serial Bus) serial communication standard into at least one command based on an ATA (AT Attachment) parallel communication standard, said conversion means includes a conversion controller operating in accordance with the program stored in said ROM for converting the at least one command and data based on the USB standard into the at least one command and data based on the ATA standard, and said access means includes an ATA controller for controlling the access to said removable storage medium based on the ATA standard.

4. An external storage accessed by a computer, comprising:

conversion means configured to be connectable to a serial communication terminal of the computer and including a ROM storing a program for receiving a first command and data based on a first standard supplied from said computer and converting the received command and data into a second command and data based on a second standard different from said first standard said conversion means automatically converting the first command and data into the second command and data under control of said program stored in said ROM; and

access means for controlling access to a storage medium removably connected to said conversion means according to the second command and data supplied from said conversion means.

5. An external storage according to claim 4 wherein said first standard is a USB (Universal Serial Bus) standard and said second standard is an ATA (AT Attachment) standard.

6. An external storage according to claim 4 or 5, wherein said conversion means includes means for converting serial data of format based on the first standard and supplied serially from said computer into parallel data of format based on the second standard, said access means includes means for writing the data of the format based on said second standard into said storage medium in response to a write command based on said second standard supplied from said conversion means, said access means reads the data stored in said storage medium in response to a read command based on the second standard supplied from said conversion means and supplies the data to said conversion means in accordance with a format based on said second standard, and said conversion means includes means for converting the data supplied from said access means into serial data of format based on said first standard and supplying the data thus converted to said computer.

7. An external storage according to claim 4 or 5, wherein said conversion means includes means for sending a response to said computer without converting a command, which is included in commands supplied from the computer and which is capable of responding to the computer without using the access means, into a command based on the second standard, and for supplying a command, which is included in the commands supplied from the computer and required for accessing the storage medium by means of the access means, to the access means with said command being converted into a command based on the second standard.

8. An external storage according to claim 4 or 5, comprising mounting means for mounting said storage medium removably, wherein said access means accesses said storage medium mounted on said mounting means.

9. An external storage according to claim 8, wherein said mounting means includes a plurality of mounting members for selectively mounting a plurality of removable storage media.

10. An external storage according to claim 8, wherein said storage medium includes a flash memory, and said external storage functions substantially in similar fashion to a magnetic disk drive.

11. An external storage according to claim 4 or 5, comprising mounting means for mounting said storage medium removably, wherein said access means is arranged in said storage medium mounted on said mounting means.

12. An external storage according to claim 11, wherein said mounting means includes a plurality of mounting members for selectively mounting a plurality of removable storage media.

13. A converter system comprising:

a first node based on a serial communication standard;

a second node based on a parallel communication standard;

conversion means for converting a command based on the serial communication standard supplied serially through said first node into a corresponding parallel command based on the parallel communication standard and outputting said parallel command to said second node in the case where said command based on said serial communication standard is a command requiring access to a system based on said parallel communication standard; and

transmission means for transmitting through said first node a response to a command based on the serial communication standard supplied serially through said first node without converting said command into a command based on the parallel communication standard in the case where said command based on said serial communication standard is a command not requiring access to said system.

14. A converter system according to claim 13, wherein said serial communication standard is a USB (Universal Serial Bus) standard, said parallel communication standard is an ATA (AT Attachment) standard, and said system includes an external storage based on the ATA standard.

15. A converter system according to claim 13 or 14, wherein said conversion means includes means for converting the format of data based on said serial communication standard supplied through said first node into the data of the format based on said parallel communication standard and outputting said data to said second node, and means for converting data of the format based on said parallel communication standard supplied through said second node into data of the format based on said serial communication standard and outputting said data to said first node.

16. A converter system according to claim 13 or 14, wherein said first node is connected to a serial communication terminal based on said serial communication standard of a computer, and said second node is connected to access means for accessing a storage medium based on a command supplied from said conversion means.

17. A converter system according to claim 16, wherein said second node is fixedly connected to said access means, and said access means is connected to means for accessing a removably-mounted storage medium.

18. A converter system according to claim 16, wherein said second node is removably connected to said access means, and said access means accesses said storage medium while being connected to said second node.

19. A converter system according to claim 13 or 14, wherein said conversion means and said transmission means include a memory for storing a program for converting a command based on the USB (Universal Serial Bus) standard into a command based on the ATA (AT Attachment) standard and a program for responding to a command based on the USB standard, and a processor for receiving a command supplied through said first node and executing a program corresponding to said received command thereby to convert a command or respond to a command.

20. A recording medium for storing a program including an instruction for instructing a processor to convert a command based on a USB (Universal Serial Bus) standard into a command based on an ATA (AT Attachment) standard, an instruction for instructing the processor to respond to a command based on the USB standard, an instruction for instructing the processor to convert data of a format based on the USB standard into data of a format based on the ATA standard, an instruction for instructing the processor to convert data of the format based on the ATA standard into data of the format based on the USB standard.

* * * * *